

INTERNATIONAL APPLE CORE
REGION 2 DIRECTOR
HARLAN FELT
359 LAWTON ROAD
RIVERSIDE, IL 60546
312-447-6267

Welcome to the International Apple Core. Enclosed is the first mailing to member user groups from your Regional Directors.

I am your representative to the IAC and I can be reached at the above address or phone number if you have any questions about the organization. Upcoming mailings will include such things as APPLE AP-NOTES, technical hints from user groups, software from the APPLE user contributed software bank and much more.

One of the major purposes of the IAC is to provide an interface and an interaction between the APPLE user groups and APPLE COMPUTER INC. A formal organization such as the IAC is the best way to accomplish this relationship.

If you know of any other APPLE user groups in your area, please pass on the existence of the IAC and have them contact me. (a blank application is enclosed for this purpose). We would like to see the IAC representing as many APPLE user groups as possible.



INTERNATIONAL APPLE CORE

Dear Member Clubs:

On March 13 representatives of 60 member clubs met in San Francisco to formally initiate the International Apple Core. Directors were elected, lengthy discussions were held, and policies were worked out. Directors and officers met for the following two days arranging quite a few important details. You will receive a detailed report of the proceedings as soon as possible. In the meantime this brief summary should answer many of your more common questions.

The International Apple Core (IAC) will be a non-profit organization dedicated to the exchange of information among Apple Computer users. This will be broadly interpreted to include technical information, software, programming information, and anything else which can benefit Apple users throughout the world. In addition the IAC will provide for communications between members and the various product manufacturers. Specific areas of IAC activity will include publication of the "Orchard", maintenance of a public-domain software library, support of special interest groups, support of software and hardware standards, technical support, promulgation of ethics, and organization of annual Apple - faires.

Being an international organization, the IAC seeks to represent Apple users everywhere. Accordingly, the provisional constitution and by-laws adopted by the representatives in San Francisco allow for two directors from each of four regions in the United States (see map). In addition each continent will have one director. Since international directors may have difficulty attending conferences and meetings in the U.S., they may designate a stateside director to communicate with them and act as their proxy. Directors serve two - year terms. However in the first year one from each region serves a one - year term to allow staggered elections. Term length was assigned by coin toss. All directors will be elected by the member clubs in their areas, on a one-club, one-vote basis. Each region's director's will be contacting members to arrange meetings and a representative method of voting.

The IAC does not intend to have individual memberships. Only non-profit Apple user groups may be members. Educational institutions and other interested non-profit organizations may become associate members. They will be entitled to all the free printed information the IAC provides its members, but can not vote for directors. Commercial enterprises may become sponsors,



INTERNATIONAL APPLE CORE

entitled to information and participation in standards establishment. They, too, would be non-voting, but would receive preferential advertising treatment in IAC publications. Dues for members and sponsors will be \$50 and \$200, respectively. They will be collected every January First. Associate members pay no dues.

Officers will be a President, Vice - President, Secretary, and Treasurer. To avoid concentration of power, the bylaws provide that a director may not also be an officer. Officers are appointed by the Board.

We currently are polishing up the latest draft of the bylaws and constitution. After they've been cleared by attorneys every member will receive a copy for comment. After a suitable period a final version will be prepared, subjected to approval, and submitted to the State of California for incorporation.

The "Apple Orchard" will be the IAC's official magazine. The Board resolved to make the "Orchard" the definitive publication on the Apple Computer. To this end, Val Golding has been retained as Editor. He will quit his present job to take up editing full-time. The Board is also seeking a professional publishing house to handle subscriptions, printing, and mailing. The next issue will be published September First, with subsequent issues following quarterly. The publication rate should go up to monthly as we gain experience. Articles should be submitted to Val Golding, and should clearly indicate that they're for the "Orchard." Preliminary submission deadline for the first issue is July First. The "Apple Orchard" will replace future editions of "Contact." Unlike "Contact," the "Orchard" will be available either by subscription or by sale through clubs or stores. Apple Computer Company intends to purchase some "Orchards" from the IAC. One copy will be sent to each newly - registered Apple owner to tell them about user groups.

A number of committees and special interest groups were established to deal with specific subjects. A list of these committees, their chairmen, and their functions is attached. You can contact any of them if you have interest. The IAC will try to establish a hot-line for technical and software problems. Until permanent arrangements can be made, you will have to continue using current lines provided by various clubs and Apple Computer, Inc. Neil Lipson's software committee will be



INTERNATIONAL APPLE CORE

collecting, documenting, and distributing a diskette a month to the member clubs. These will contain public domain software and will be distributed free of charge. The IAC will also distribute to members application notes from Apple Computer and other suppliers as they are made available to us.

The IAC's first letter to clubs indicated that the "Source" telecommunications network would be made available. For a variety of reasons that did not work out as we had hoped. We are currently negotiating with the "Source", "Micronet", and others to make some form of telecommunications possible. You will be notified as soon as something is arranged.

Obviously a large number of questions are still unanswered. I will be trying to get all the information I can to you in the near future through various communications. In the meantime any of the officers or directors will be more than happy to answer your inquiries. Please bear with us over the coming months as we put the IAC together and shake out the bugs. Most importantly, please communicate with us if you have any suggestions about how something should be done, or could be done better. This is your club; only you can make it work.

Regards,

Joe Budge
Secretary,
International Apple Core

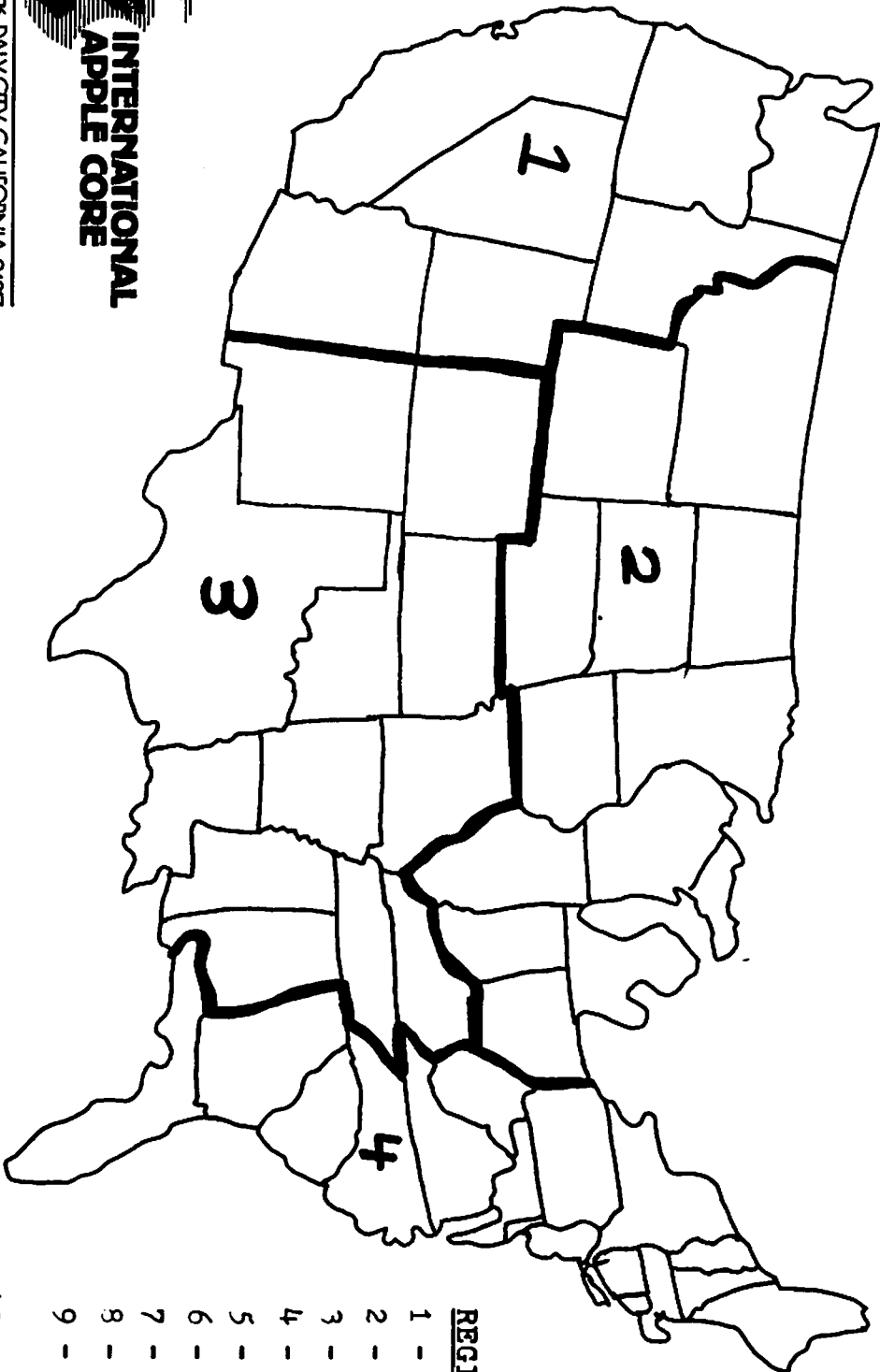
jb:JHB

4 Enclosures



**INTERNATIONAL
APPLE CORE**

PO BOX 976 DAVY CITY, CALIFORNIA 94017



REGIONS:

- 1 - WEST
- 2 - NORTH
- 3 - SOUTH
- 4 - EAST
- 5 - AFRICA
- 6 - ASIA
- 7 - AUSTRALIA
- 8 - EUROPE
- 9 - N. AMERICA
(ex U.S.A.)
- 10 - S. AMERICA



INTERNATIONAL APPLE CORE

DIRECTORS

MARCH 13, 1980

One-Year Terms:

1 - WEST

Joe Alinsky
22240 Wyandotte
Canoga Park, Calif. 91303
(213)-703-1894

2 - NORTH

Jon R. Lawrence
15487 Minock
Detroit, Mich. 48223
(313)-534-2433

3 - SOUTH

Jerry Vitt
6906 Waggoner Pl.
Dallas, Tx. 75230
(214)-369-7660

4 - EAST

Bernie Urban
6205 Walholding Rd.
Washington, D.C. 20016
(301)-229-3458

Two-Year Terms:

Fred Wilkinson
P.O. Box 40031
San Francisco, Calif. 94110
(415)-585-2240

Harlan G. Felt
359 Lawton Rd.
Riverside, Ill. 60546
(312)-447-6267

Scott Knaster
5425 E. Kentucky Ave.
Denver, Colo. 80222
(303)-355-2379

Tony Cerreta
55-A Locust Ave., #4G
New Rochelle, N.Y. 10801
(914)-636-3417

INTERNATIONAL

7 - AUSTRALIA

Neil Bennett
55 Clarence St.
Sydney, Australia 2000
612-293753

9 - NORTH AMERICA

Auby Mandell
409 Queen St. W.
Toronto, Ontario
Canada M5V2A5
(416)-868-1315

8 - EUROPE

Wolfgang Dederichs
Auf Drenhausen 2
4320 Hattingen
West Germany
02324/67412

P.O. BOX 976 DALY CITY, CALIFORNIA 94017



INTERNATIONAL APPLE CORE

OFFICERS

MARCH 13, 1980

PRESIDENT:

Ken Silverman
3673 Bassett Ct.
South San Francisco, Calif. 94080

(415)-878-9171

VICE-PRESIDENT:

Michael Weinstock
64 Pinedale Rd.
Hauppauge, L.I.
New York 11767

(516)-360-0988

TREASURER:

Dave Gordon
19273 Kenya St.
Northridge, Calif. 91324

(213)-384-0579

SECRETARY:

Joe Budge
2507 Elderwood Ln.
Burlington, N.C. 27215

(919)-228-6055



INTERNATIONAL APPLE CORE

COMMITTEES:

The following are the chairmen or organizers of the various IAC committees and special interest groups. They are just forming and need your participation. Members of member clubs, sponsors, and associate members are welcome. If you have any interest contact the chairmen directly or through the IAC. They can tell you exactly what the committee is up to. If you wish to form or participate in a special interest group not listed here, let the IAC know so that a SIG can be set up.

Orchard Publication: Val Golding, 6708 39th Ave. S.W., Seattle, Wash. 98136. (206)-932-6588

IAC Software: Neil Lipson, 29 S. New Ardmore Ave., Broomall, Pa., 19008 (215)-356-6183

New Club Assistance: Dick Sedgewick, 100 Horne St., Dover, N.H., 03820 (603)-742-3703

Constitution & Bylaws: Ken Silverman, c/o International Apple Core (415)-878-9171

Telecommunications: Craig Vaughn, 2633 E. 28th Ave. STE 622, Signal Hill, Calif. 90806 (213)-595-6858 TCA 099

Standards: Mark Robbins, 2726 S. Moline Ct., Aurora, Co. 80014 (303)-755-6440, (303)-696-0200

Newsletter Librarian: Major Terry N. Taylor, 12319 E. Bates Circle, Aurora, Co., 80014 (303)-750-5813

Newsletter Exchange Coordinator: David Alpert, 880 Mellody, Lake Forest, Ill. (312)-295-6078

Education SIG: Ted Perry, 5848 Riddio, Citrus Hts., Ca. 95610 (916)-961-7776

Handicapped SIG: Bernie Urban, 6205 Walholding Rd., Washington, D.C., 20016 (301)-229-3458

Legal SIG: Butch Clayton, P.O. Box 70278, Charleston Heights, S.C., 29405 (803)-884-5370 (803)-554-9171

Ham Radio SIG: James E. Hassler, 129 Park Ave., Orchard Valley, Cheyenne, Wy. 82001 (307)-632-4934, WB7TRQ on 14.329Mhz
Sunday Nights at 5, PST.



INTERNATIONAL APPLE CORE

APPLICATION FOR MEMBERSHIP

Name of Organization: _____

Class of Membership: _____

Mailing Address:

Street: _____

City: _____ State: _____ Zip: _____

Country: _____

(If the above is a post office box, please supply a street address
below where parcels may be sent:)

Officers:

NAME

PHONE

President: _____

Treasurer: _____

Editor: _____

Other: _____

Terms Expire: _____ Copies of "The Apple Orchard" desired

Number of Members: _____ at \$ 1.00 each: _____

Total Remittance Enclosed: \$ _____

- Full membership is available only to Apple User's Groups. A \$50.00 initiation fee must accompany this form. Only full members may order the "Orchard" with this form.
- Applicants for Associate Membership are asked to provide evidence that they are a non-profit institution. There is no membership fee.
- Sponsors: Please indicate the name, position, and telephone number of the person in your organization responsible for liason with the IAC. The Sponsoring Membership fee is \$200.00



INTERNATIONAL APPLE CORE

Dear IAC Member Club:

At the first annual meeting of the IAC many subjects were discussed like special interest groups, by-laws, and information transfer. A full report should be forth coming dealing with the meeting.

I would like to inform you of the outcome of certian subjects in which there has been a great deal of concern - those subjects being 1) our publication "The Apple Orchard", 2) Club Dues, 3) The Source, and 4) Software.

1) "THE APPLE ORCHARD" - It was decided, at this time, to have a quarterly issue starting in September of 1980. If it is accepted and is a success it could be published every other month. The cost on the news stand will be \$3.50. The cost to member clubs on a bulk purchase will be \$2.00 a copy. If an individual subscribes it will be \$10 for four issues - or \$2.50 a copy. We recommend individual subscriptions to speed delivery to the end Apple user. If mailed to a club it would take time and money from that club for re-distribution. The method in which subscriptions will be handled is forthcoming - we are in the process of signing agreements with the publishers.

2) CLUB DUES - The Board of Directors decided that each member club will pay an annual dues of \$50. This will start in January of 1981. A new club will still pay the \$50 initiation fee in addtion to the annual dues. The membership size of a club will have no bearing on the dues cost. It is basically a token amount so that a club shows an interest in the IAC. The IAC will most likely send each club a free software disk each month - this alone makes up the \$50 dues.

3) THE SOURCE - Originally we informed some of our member clubs that a free "Source" account would be forthcoming so that there would be open communications between clubs and the IAC. The "Source" has and still is going thru some major changes. We are still in contact with them and are in the closing stages of an agreement. When the agreement is signed our member clubs will be informed. The following is what we hope will come out of the negotiations:

- a) IAC Source mail box for clubs to leave messages & questions.
- b) Individual accounts for the officers and directors to conduct IAC business.
- c) A club account for each member club. The account fee (\$100) would be FREE but there would be monthly billing for usage at Source rates. One member would have to be responsible for the club account by putting the billing on his/her VISA or Master Charge card. Who or how that club account is used will be up to the club and the individual responsible. In addtion the necessary software to use the Source with the Apple will be supplied FREE. All club account numbers will be published in the Orchard.


P.O. BOX 976 DALY CITY, CALIFORNIA 94017

4) SOFTWARE - It was agreed that software being supplied by the IAC to member clubs will be FREE, even the disk it is supplied on will be FREE. We hope to give a "Disk a Month" to each member club. How a member club gives it to its membership will be up to them. The programs on each disk can not be charged for but a club my charge for the "medium". Distribution will be by the Software Committee.

NOTE: At the meeting in March two disks were given to each Director to be given to the clubs in their area. One disk had some new utilities and the other was for owners of an Apple II Plus, which contains a good mini-assembler and it has a version of INTEGER Basic but it won't work in all program cases.

I hope this will answer some of your more immediate questions but if you still are concerned on any subject please don't hesitate to call me in the evening. I can be reached on 415+878-9171 between 5:30 and 10:30PM Pacific time.

Thank You,


Ken Silveman, President



TO: Member Clubs
Sponsors
Associates

Dear Members:

Enclosed you will find the complete minutes of the International Apple Core's meetings to date. The first meeting was held in San Francisco last October. This was the start-up meeting: all the attendees were flown in by Apple Computer and asked to form an International Users Organization. After the meeting the various officers, directors, and chairmen went to work contacting clubs, arranging publication of The Orchard, and drawing up the outline of the IAC. The next meeting was held in March, when the representatives of member clubs convened. The first regular directors and operating officers were elected at this meeting, and most of the IAC's structure was worked out. Following the Annual Meeting of the clubs, the new Board of Directors met to begin organizing the administration and policies of the IAC. A second meeting was held in May to continue this organizational work. Most directors were then in Anaheim, California as guests of Apple for the unveiling of the Apple III.

I am sure you will find the minutes of these meetings quite interesting and informative. Many of our new members will be learning the detailed history of the IAC for the first time. Older members may have attended some of these events, but few have attended all. Therefore, these minutes will explain many details of the IAC which haven't yet been generally distributed.

As always, please feel free to contact your directors or officers if you have any questions about this material.

Regards,

Joseph H. Budge
Secretary
International Apple Core

INTERNATIONAL APPLE CORE

BOARD OF DIRECTORS MEETING MINUTES

Submitted by Ken Silverman - Secretary

DATE: October 27, 1979
PLACE: Ramada Inn, San Francisco
ATTENDIES: See attached list

The meeting was opened at 9 a.m. by Val Golding of the Apple Puget Sound club. Everyone introduced themselves (see list). Opening remarks included the reason for the meeting - i.e. the formation of an International Apple users group. Interim officers were appointed so that the business of forming an organization could be started. The following were the results of the appointments:

President - Val Golding
Vice President - Neil Lipson
Secretary - Ken Silverman
Treasurer - Dave Gordon

A name for the organization was presented and accepted as the "INTERNATIONAL APPLE CORE".

During the opening remarks and appointments two representatives of the Apple Computer Company, Phil Roybal & Jim Hoyt, helped in organization and direction. The Apple Corp. was instrumental in calling this group together, in fact, they paid to fly the representatives to San Francisco, so that this type of organization could be formed. (NOTE: It was talked about and confirmed several times during the conference that Apple would help only in getting the group started. The organization is to be completely independent of all manufacturers and software companies and is responsible only to its user membership).

Many items were presented by attendees and a list of subjects to be covered were put on display paper by Phil Roybal so we could detail them as needed. Items to be covered:

I - THIS GROUP

- 1-Today
- 2-March West Coast Computer Faire
- 3-Task Forces
- 4-Local user group problems

II - USER GROUP OBJECTIVES

- 1-Funding
- 2-Communications
- 3-Special interest groups
- 4-How to form a user group library
- 5-Bylaws and purpose
- 6-Local/National user interaction

- 7-Mfg. interface
- 8-Membership
- 9-Social goald

III - MARCH MEETING OUTLINE

- 1-Attendees
- 2-Logistics
- 3-Apple Faire

IV - NON-AGENDA ITEMS

At this time the items in group II were discussed as items from TODAY in group I:

It was brought up that unless this group could communicate freely in all directions it would be difficult to obtain credibility. Communications must be open to all member clubs in both directions and to all hardware and software companies - so that information can be shared. This will be accomplished by the use of the network called the SOURCE. Apple & Joe Alinsky will get us a free account (not to be abused) for our use. Randy Hyde will get the account and user info to member clubs and board members as soon as it is approved.

NOTE: AS THE DAY WENT ON COMMITTEES WERE FORMED AND CHAIRPERSONS APPOINTED - SEE ATTACHED LIST.

At several times during the weekend funding was discussed. In order to bring in membership (both clubs and its members) we needed to offer them something, a "CARROT". The Apple Corp. said they had a large pile of application notes (something like a Woz pak) that they could give out. In addition a complete set of all Apple reference manuals, card manuals, etc. would also be given. These items and a cover letter by the International Apple Core would serve as an inducement for the clubs. The club would have to pay a one time initiation fee of \$50 to be a member of this organization. Apple will mail this package to their complete list (around 75) of Apple user groups. Bernie Urban said he would look into Federal Funds. An annual dues structure would be discussed at the first annual meeting after we signed up member clubs. (Dave Gordon - interim Treasurer said this would most likely be around \$5.00 per person a year).

The talk of dues and club membership brought up the subject of types of membership. Three types were discussed:

- User Group Membership
- Associate Membership - Non-profit & non-voting
- Sponsoring Membership

The Sponsoring Membership was created for commercial business like Apple Corp. and other manufacturers. It would cost them \$1000 a year to be a member or if they joined now before the March Faire, \$750. For their membership they would get the following:

Choice of Ad space in publication
New product feedback
Application notes
Member of standards committee
Publication subscription
Right of input
Right of response
A vote on the standards committee

A Sponsor would have no vote on the Board of Directors in the organization.

The associate membership would be for organizations like schools (not non-profit users groups). They would have no fee and no vote but would receive a copy of the publication and other information whenever possible.

The SPECIAL INTEREST GROUPS topic was discussed with the end result as we would support all of them. This would be more on a local level because there are so many of them. A STANDARDS & CONVENTIONS Committee will be formed during the March meeting to be held in 1980. The formation of a committee to put a CLUB KIT together was completed to be chaired by Dick Sedgewick. This committee will compile information on how to start a club, a library, set up communications with the International group and much more.

At this time one of the Task Forces was formed. This group would put out a major publication to coincide with the West Coast Computer Faire and consist of input from all the large clubs via their editors. The publication is expected to be over 100 pages long and will retail for about \$5.00 a copy. All member clubs will get copies for their membership for \$1.00 a copy. This issue will most likely cost over \$60,000, over 50,000 copies will be made, and we hope to bring in \$100,000 (see attached sheet on newsletter funding). Apple Corp. will also mail a copy to everyone on their Apple owner list (they will pay for the copies). Val Golding, editor of Call Apple will head this group (see list for other members of this committee). All copy for this issue must be in Val's hands by December 1st. At this time Larry Wise, of Information Unlimited, who was sitting in on the meeting offered copies of the word processor "EasyWriter" to each editor on the committee. This was accepted and the board in return would like to give Information Unlimited a 1/2 page in the publication. Motion was made and carried. Larry thanked the board.

A discussion of the March meeting was then taken up. It was decided to hold our meeting on Thursday before the Friday opening of the Faire (March 13th). At this meeting votes would be taken on the following:

Constitution & Bylaws
New Officers
Fees
Committees
Confirmation of the Board of Directors

During the Faire, on Saturday, we would summarize the organization and have seminars (See list for Faire committee).

The day was getting on and at this time we listed the committee responsibilities

Business plan

Action items

Expenses - Budget

Justification of committee

Each committee was asked to work on the above and to submit as soon as possible.

The meeting adjourned at 4:35.

INTERNATIONAL APPLE CORE

BOARD OF DIRECTORS MEETING MINUTES

Submitted by Ken Silverman - Secretary

DATE: October 28, 1979

PLACE: Ramada Inn, San Francisco

SECOND DAY OF CONFERENCE

The meeting opened at 9:06 by Val Golding interim President. It was stated that most attendees had flights out of San Francisco early.

The topics for today:

I - Board of Directors:

- 1 - Where to meet
- 2 - How is Core to be organized
- 3 - How to insure geographical distribution
- 4 - Pay board member expenses

The following is the results of discussions on how the Board Of Directors should be formed and who has a vote on that board:

There will be four regions in the United States -

North - South - East - West

These would be divided by:

The Mason-Dixon line in the South

The Rockies in the West

The Appalachians in the East

Each region would have two representatives, voted by the members in their area, who would serve on the Board of Directors. There also would be two representatives from outside of the United States. This gives a total of 10 votes - these 10 directors then would elect a President of the Core and that person would also have a vote on the board. The President would not also be an area representative. This gives a total of 11 votes. The President and Board would then elect the rest of the officers and committee persons. The rest of the officers and committee persons would not have a vote on the board.

The term of office for an area board member would be two years except in this first year to start in March 1980. In this first year one of the representatives from an area would serve a one year term and the other a two year term. This would give the organization continued continuity thru the years. The officers, including the President, and committee person would serve a one year term but is re-newable.

If for any reason an area representative cannot attend an annual meeting they may vote on agenda items by proxy.

Much more in this area needs to be done and the draft of the bylaws should cover most of them.

Ken Silverman suggested that the International Apple Core logo be a globe of the world with a bite taken out and a crescent moon over it - motion made and carried - Apple will have their graphic department do drawing.

Before adjourning the topic of ETHICS was discussed. The following policy was adopted by the members present:

We do not condone the piracy of software.

No club library will include copyrighted software unless it is released for duplication by the author.

The meeting will be held on Thursday March 13th in San Francisco - time and place to be forthcoming.

If enough monies come in via the clubs and the sponsoring memberships the Core will try to pay for the area representatives to fly to the meeting.

Meeting ended at 11:30 a.m.

INTERNATIONAL APPLE CORE
SPECIAL FAIRE NEWSLETTER FUNDING

SOURCE	COPIES	AMOUNT
ADs		\$24,000
Apple Corp	25,000	25,000
SF Apple Core	1,000	1,000
A.P.P.L.E.	4,000	4,000
Applesause	1,000	1,000
ABACUS	300	300
Houston	400	400
N.E.A.T.	150	150
Apple Pie	250	250
Michigan	250	250
Phila	150	150
Stores	15,000	45,000
Faire	3,000	7,500
<hr/>	<hr/>	<hr/>
	45,5K	\$108K

SOFTWARE DISSEMINATION
Plan 1/1

Neil Lipson - Chairman
Randy Hyde
Larry Danielson
Ed Avelar
Kip Reiner

INFOR TRANSFER

Randy Hyde - Chairman
Bernie Urban
John Lawrence
Mat McIntosh
Mike Weinstock
Val Golding
Jim Hoyt
Ken Silverman

KIT COMMITTEE

Dick Sedgewick - Chairman
Bernie Urban
Fred Wilkinson
Bob Collins
John Lawrence

INTERIM OFFICERS & COMMITTEES

INTERIM OFFICERS

PRESIDENT	Val Golding
VICE PRESIDENT	Neil Lipson
TREASURER	Dave Gordon
SECRETARY	Ken Silverman

INTERIM REGION DIRECTORS

NORTH	John Lawrence ?
SOUTH	Bob Collins Dewayne VanHoozer
EAST	Bernie Urban Dick Sedgewick
WEST	Fred Ilkinson Joe Alinsky
INT	Roger Ossie ?

COMMITTEES

NEWSLETTER Articles by 12/1	Val Golding - Chairman Ken Silverman Randy Hyde Ed Avelar Mark Crosby (Washington DC) Ed Seeger
THE SOURCE	Joe Alinsky - Chairman Craig Vaughn Jim Hoyt
FAIR - LOGISTICS Speaker call 11/3 Budget 12/1 Schedule 12/1	Mat McIntosh - Chairman Fred Wilkinson Dave Gordon Phil Roybal
CONST. & BYLAWS Draft 1/30	Ken Silverman - Chairman Jim Hoyt
CARROT Ship 11/15	Randy Hyde - Chairman Jim Hoyt
BUDGET/FINANCE Input 1/1 To Sec 1/30 Mail 2/1	Dave Gordon - Chairman Val Golding Mike Weinstock Rudge Allen

PERSONS PRESENT AT 10/27/79 MEETING

NAME	ADDRESS	CITY	ST	ZIP	PHONE
Ken Silverman	3673 Basset Ct.	S. San Francisco	CA	94080	415+878-5382
Neil Lipson	29 S. New Ardmore Ave.	Broomall	PA	19008	215+356-6183
Kip Reiner	19041-1 Hamlin	Reseda	CA	91335	213+876-6600 x 150
Jon Lawrence	15487 Minock	Detroit	MI	48223	313+534-2433
Fred Wilkinson	P. O. Box 40031	San Francisco	CA	94110	415+585-2240
Joe Alinsky	22240 Wyandotte	Canoga Park	CA	91303	213+703-1894
Dave Gordon	19273 Kenya St.	Northridge	CA	91324	213+384-0579
Gary Koffler	13947 Oxnard #109	Van Nuys	CA	91401	213+787-3890
Bernic Urban	6205 Walhonding Rd.	Washington	DC	20016	301+229-3458
R. V. Bob Collins	12502 Bexley	Houston	TX	77099	713+495-3777
Dewayne VanHoozer	5310 Lost Forst #154	Houston	TX	77292	713+864-1654
Larry Danielson	5302 Camino Alto Mira	Castro Valley	CA	94546	415+581-2748
Ed Avelar	2850 Jennifer Dr.	Castro Valley	CA	94546	415+538-2431
Dick Sedgewick	100 Horne St.	Dover	NH	03820	603+742-3703
Matthew McIntosh	534 Jessie	San Francisco	CA	94101	415+552-9234
Jim Hoyt	10260 Bahdley Dr.	Cupertino	CA	95030	408+996-1010
Val Golding	6708 39th Av. SW	Seattle	WA	98136	206+932-6588
Randall Hyde	12804 Magnolia	Chino	CA	91710	714+682-5268
Phil Roybal	1111 Pippin Creek Ct.	San Jose	CA	95120	408+268-7939

INTERNATIONAL APPLE CORE
ANNUAL MEETING OF THE FULL MEMBERSHIP

MINUTES

Submitted by Joseph H. Budge - Secretary
March 13, 1980

The first Annual Meeting of the Full Membership of the International Apple Core, herein after referred to as the "General Meeting", was held at 9:00 AM on Thursday, March 13, 1980 in the San Francisco Convention Center in San Francisco, California. Mr. Silverman, the interim Secretary, was present as the presiding officer. Proceedings were videotaped by Mr. Alinsky in lieu of a recording officer.

ATTENDANCE:

As the General Meeting does not constitute a deliberative assembly, determination of a quorum was not necessary. A list of member clubs represented is attached (Appendix I).

BUSINESS:

The interim Secretary stated that the areas of representation had been reapportioned by the officers to follow state lines instead of cutting across states. A map was displayed and approved by the assembly (Appendix II). Following this the Secretary showed how the interim Board had structured the IAC so that each area would elect Directors who in turn would appoint Officers (Appendix III).

TEMPORARY ADJOURNMENT:

At 9:30 AM the General Meeting was adjourned for one hour. During the temporary adjournment the Full Members caucused to elect Directors. Following their election the Directors met to appoint the Officers. The General Meeting resumed at 10:30 AM.

INTRODUCTION OF DIRECTORS AND OFFICERS:

The elected Directors were introduced and recognized by the assembly. The Directors were:

JOE ALINSKY
JON LAWRENCE
JERRY VITT
BERNIE URBAN
NEIL BENNETT

FRED WILKINSON
HARLAN FELT
SCOTT KNASTER
TONY CERRETA
WOLFGANG DEDERICH

The following officers were appointed to one - year terms by the Board:
PRESIDENT: KEN SILVERMAN

VICE PRESIDENT: MICHAEL D. WEINSTOCK
SECRETARY: JOE BUDGE
TREASURER: DAVID GORDON

More detailed descriptions of the Directors and Officers are attached (Appendix IV).

REPORTS OF INTERIM OFFICERS:

The interim President, Mr. Golding, reported that the original idea of the IAC had been to serve Apple user clubs. He was very pleased with the results so far and with the way in which the IAC idea had been accepted. The first issue of the "Orchard" had been published. As this was a rushed issue, he expected future issues to be of even better quality.

The Treasurer reported that the IAC held \$17,000 in cash at the bank. His complete report is attached (Appendix V). After discussion it was determined that the Treasurer will become bonded.

The outgoing interim Officers, Mr. Golding, President, and Mr. Lipson, Vice President, were thanked for their help and efforts.

REPORTS OF COMMITTEE CHAIRMEN:

The Bylaws committee reported that a first draft of the Constitution and Bylaws had been prepared. The Constitution spelled out the organization's rights and restrictions. It would be used as the Charter in incorporation and therefore would be difficult to change. The Bylaws, on the other hand, provided flexibility for the organization. They could be changed under the rules set forth for amendment. Both Constitution and Bylaws were put on the floor for discussion later during the General Meeting.

The Software Committee reported that it had organized a tree system for the distribution of software. In answer to a question from the floor it was explained that the members of all committees at that time were volunteers from the first organizational meeting of the IAC. The Software Committee did not intend to distribute every public domain program as there were too many. Instead it would try to distribute the best and make available a list of who held what public domain programs for exchange purposes.

The Kit Committee had proceeded towards defining a kit for forming clubs. The kit will consist of a sample Constitution and Bylaws, a description of how to start a club, and some software to help start the club's library.

DISCUSSION:

The majority of the General Meeting consisted of explanation and discussion of the IAC's organization and operations. These discussions are summarized below:

A large part of the discussion centered around the Constitution and Bylaws. The Constitution was explained and left largely unaltered. The Bylaws were re-written during the course of the discussion to reflect technical improvements and the consensus of the assembly. The Bylaws, as so discussed and modified, provide for three categories of membership: Full, Associate, and Sponsor. Full members would be clubs and would have the only vote. Associates would be non-profit institutions who would

receive printed matter free. Sponsors would be companies or individuals who would receive special advertising privileges as well as the normal flow of information from the IAC. Business of the IAC would be guided by a Board of Directors, of which there were 10 members. Two Directors would come from each region of the United States and two would come from outside the U.S. (Appendix III). The Directors would elect the Officers to provide for the daily operations of the IAC. Other provisions in the Bylaws provided for amendment and referenda of the membership. It was determined that a final draft of the Constitution and Bylaws was to be prepared. Apple Computer's legal staff had volunteered to participate in the drafts' preparation. When ready, the draft would be sent to all the Full Members for additional comments and suggested changes. After final review by the Board and addition of final changes, the documents would be used for incorporation of the IAC. Additional changes to the Bylaws would then be available under their amendment provisions. The entire process was expected to take about six months.

A questioner from the floor asked why the West Coast wasn't receiving greater representation since there were more Apple computers there than anywhere else. The chair answered that the subject had been extensively debated during the first organizational meeting. The participants finally decided that it was important to keep any one group from dominating the IAC. Therefore the equal representation arrangement was agreed to. The chair further pointed out that the Bylaws are open to change by the Board or by referendum if this arrangement is too objectionable.

Distribution of the "Orchard" was then explained. Apple Computer had bought 25,000 "Orchards" to send to registered owners on the "Contact" list. These were sent free of charge. Unfortunately Apple's mailing list only covered about 25% of all Apple owners. To close the gap, the IAC printed another 25,000 for distribution through clubs and retail outlets. These would be sold to old and new Apple owners alike. Beginning with the next issue Apple planned to send each new owner a complimentary copy of the "Orchard" by way of introducing the IAC. Orchards would be available to everyone else through subscription, through retail outlets, and through clubs.

It was also explained that Apple Computer, Inc. had sent out the package of manuals to every new member in the IAC. These had been provided free. However, Apple had announced that the packages would not be provided after the General Meeting. The Assembly thanked Apple for the packages which had been sent.

A questioner from the floor asked if "Call-A.P.P.L.E." would be merged into the "Orchard". Mr. Golding responded that it definitely would not. The editorial contents of the two publications would be entirely different.

The IAC's dissemination of information was explained by the Chair. Two types of dissemination were planned. One would be the "Orchard", subject to later approval by the Board. The other type of information would be in more specialized forms such as the distribution of software, Apnotes, and the like.

At this point Major Terry Taylor, of Denver Apple Pi, asked the audience to tell their members to stop sending him copyrighted software. Denver Apple Pi maintains a large library of public domain software and an active exchange program. Major Taylor's request generated a discussion on public domain software and the IAC's role with respect to it. Mr. Lipson, Software Chairman, said that he hoped IAC software could be donated by member clubs and be derived from articles in the "Orchard." While the software was public

domain he hoped that clubs would refrain from freely distributing it to non-members in order to promote the IAC. A member of the audience remarked that his club was upset to see its software being freely exchanged around the country. This was software the club had exchanged with another club. Finally another member in the audience commented that public domain software was just that: public and open to all.

The discussion then turned to the alternative types of membership available in the IAC. Associate members were envisioned as schools, rehabilitation centers, and other organizations that can't really afford membership, but frequently have Apples and need the information. The IAC hoped to provide them with the free information sent to clubs. Sponsors, on the other hand, would be manufacturers of software or hardware. They would be charged more for membership but would be given some benefits that clubs don't need. Advertising preference and mailing lists were cited as examples.

A question was raised on the floor over the dues that would be charged to member clubs. A letter sent to members by interim Officers in February had suggested a sliding rate based on club size, with the fee centering around \$150.00. It was explained that the letter had generated considerable response which forced re-examination of the dues structure. The sliding rate based on size would promote cheating and dishonesty and was therefore not recommended. Furthermore, that dues structure was considered excessive and impractical. If all 75 Apple clubs known at that time had joined, dues would provide \$11,250 in revenues. It was likely that such a high fee would scare off many clubs. Software sales were considered for revenue too. But if each club paid \$10.00 per diskette per month, only \$9,000 would be generated. On the other hand "Orchard" subscriptions and advertising could be expected to net \$75,000 in the first year. Since it was obvious that the IAC would have to depend on the "Orchard" for the majority of its funding, the interim Officers planned to recommend that the Board set the dues at a nominal figure. \$50.00 per year was being discussed.

A suggestion was made from the floor that the IAC fund grants in Apple-related research with any excess funds. This was generally agreeable to all and was allowed by the Constitution. Further discussion on the subject was not necessary, though, as the IAC had no excess funds at the time.

APPOINTMENT OF COMMITTEES:

Several Committees and Special Interest Groups were established at the end of the General Meeting. These committees were open to any volunteers willing to do the required job. A full list of the committee chairmen appointed at the General Meeting is attached (Appendix VI).

An Information Transfer Committee was proposed for the swift dissemination of printed information through the IAC clubs. The committee would have been a tree for copying and distributing printed matter. It died for lack of volunteers, however. Its functions were to be taken over by the Board.

A Technical Committee was proposed to provide Hot-line type service for IAC members. This committee died for lack of a volunteer to chair it.

The President and Secretary were appointed to the Bylaws Committee. The Committee's function would be to finalize the Constitution and Bylaws and organize the approval process.

A Telecommunications Committee was established. The Committee's functions are to coordinate IAC activities on the different networks, to keep abuses down, and to work out ways to exchange information via telecommunications.

A Standards Committee was established to work on hardware and software standards for Apple computers and the manufacturers who provide equipment and software for them.

A Newsletter Librarian was appointed to maintain a file of member club newsletters. In addition a Newsletter Exchange Coordinator was appointed to facilitate inter-club exchanges.

Both the Software and Kit Committees were continued. An Annual Meeting Committee was discussed. It was decided to leave arrangements for the next Annual Meeting to the Board.

Several Special Interest Groups were established. These were for Education, Handicapped, and Legal uses of the Apple.

ADJOURNMENT:

The first Annual Meeting of the Full Membership of the International Apple Core was adjourned at 3:00 PM. Following Adjournment the club representatives present had an opportunity to caucus with their Directors prior to the Board Meeting at 3:30 PM.

INTERNATIONAL APPLE CORE MINUTES

March 13, 1980

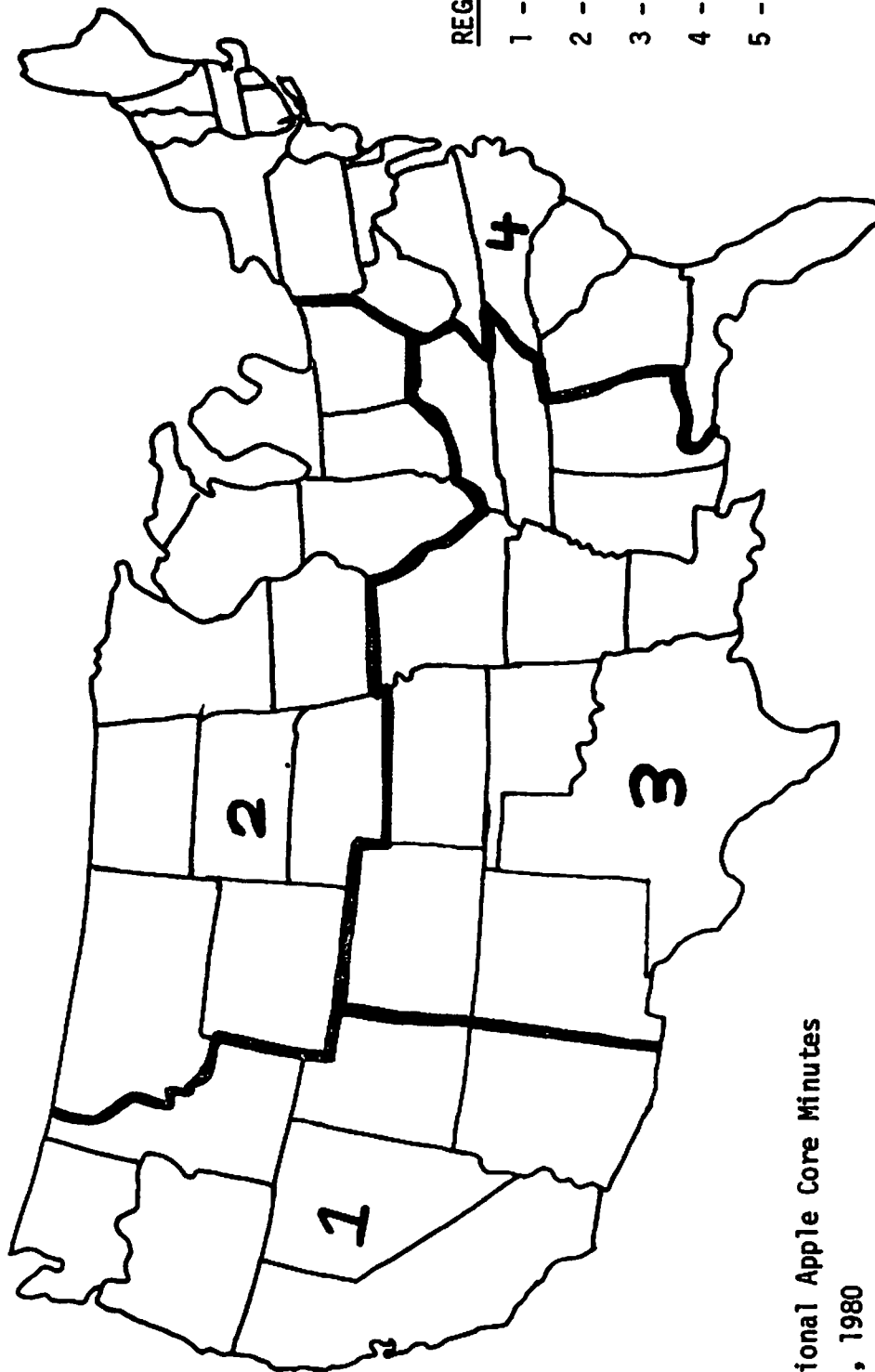
APPENDIX I

FULL MEMBERS REPRESENTED

AT THE ANNUAL MEETING

ABACUS
APPLE FOR THE TEACHER
A.C.E.S.
MICHIGAN APPLE
APPLE CORP CANADA
SAN FRANCISCO APPLE CORE
PHILADELPHIA APPLE CLUB
NORTHWEST SUBURBAN APPLE USERS GROUP
BIG APPLE USERS
DENVER APPLE PI
APPLE USERS CLUB AUSTRALIA
APPLE CORPS OF SOUTHERN NEVADA
APPLE PIE
CINCINNATI APPLE SIDER
MARYLAND APPLE CORPS
APPLE JACKS
BOSTON COMPUTER SOCIETY
HAAUGG
SANTA CRUZ APPLE GROUP
APPLE P.I.E.
GREEN APPLES
APP-LE-KATIONS

HESEA APPLE COMPUTER CLUB
ADAM II
WASHINGTON APPLE PI
ORIGINAL APPLE CORPS
CAROLINA APPLE CORE
APPLE T.A.R.T.S.
MID-HUDSON MICRO USERS
MIAMI APPLE
MIDWAY COMPUTER CLUB
SCAPPLE
APPLE USERS GROUP EUROPE
SUFFOLK APPLE COMPUTER SOCIETY
6502 USERS GROUP
THOUSAND OAKS APPLE PI
NEW ENGLAND APPLE TREE
NORTH ORANGE COUNTY APPLE CORE
A.P.P.L.E. - WASHINGTON
APPLE CORE CANADA
C.A.C.H.E.
APPLE PEELERS
APPLE CREEK



REGIONS:

1 - WEST

2 - NORTH

3 - SOUTH

4 - EAST

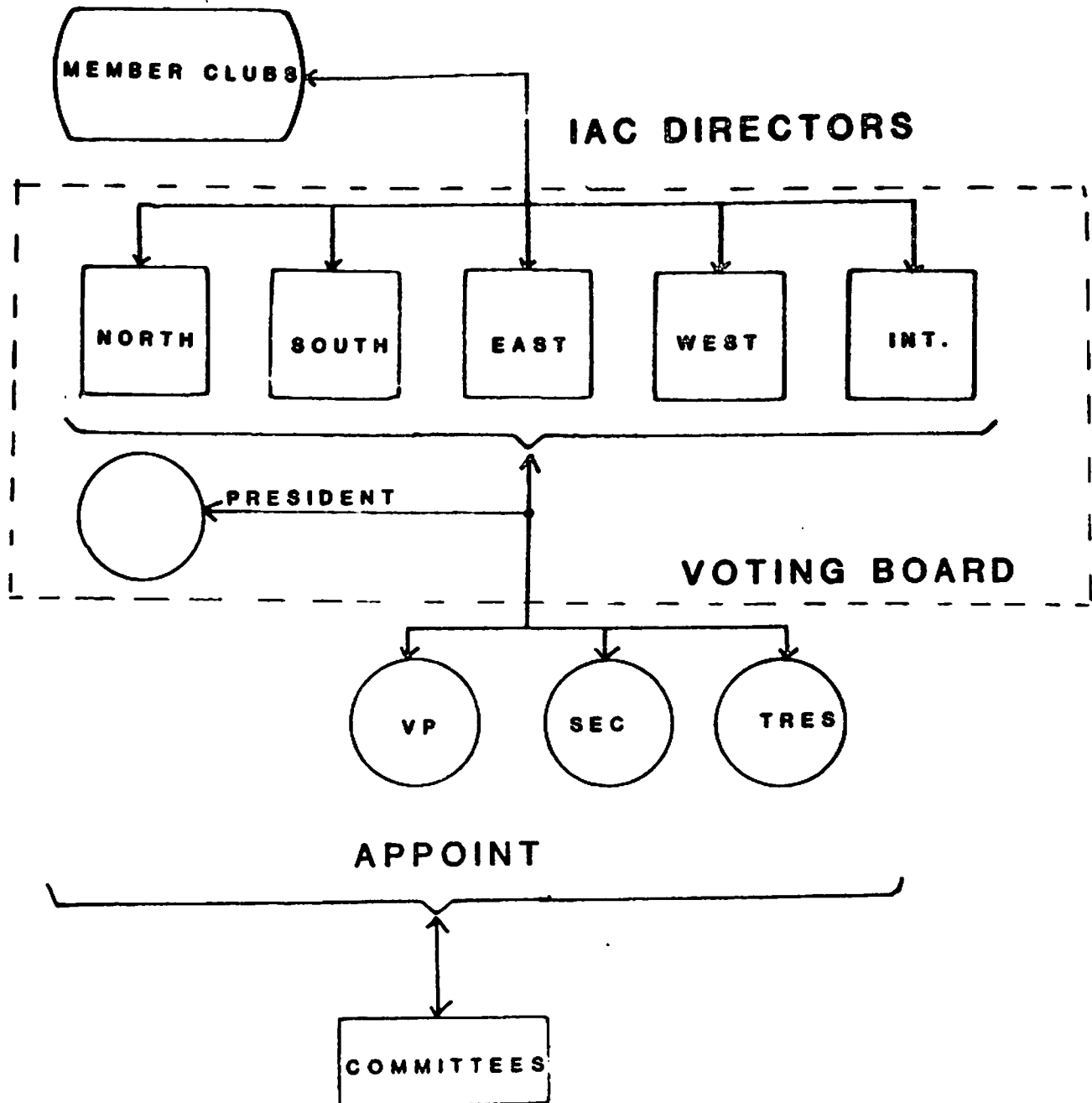
5 - INTERNATIONAL

International Apple Core Minutes
March 13, 1980
Appendix II - IAC Regions

INTERNATIONAL APPLE CORE MINUTES

March 13, 1980

APPENDIX III: ORGANIZATION CHART



INTERNATIONAL APPLE CORE MINUTES

March 13, 1980

APPENDIX IV: DIRECTORS AND OFFICERS

DIRECTORS

One-Year Terms:

1 - WEST

Joe Alinsky
22240 Wyandotte
Canoga Park, California 91303
(213) 703-1894
Original Apple Corp.

2 - NORTH

Jon R. Lawrence
15487 Minock
Detroit, Michigan 48223
(313) 534-2433
Michigan Apple Club

3 - SOUTH

Jerry Vitt
6906 Waggoner Pl.
Dallas, Texas 75230
(214) 369-7660
HAAUGG

4 - EAST

Bernie Urban
6205 Walholding Rd.
Washington, D. C. 20016
(301) 229-3458
Washington Apple Pi

Two-Year Terms:

Fred Wilkinson
P. O. Box 40031
San Francisco, California 94110
(415) 585-2240
S. F. Apple Core

Harlan G. Felt
359 Lawton Rd.
Riverside, Illinois 60546
(312) 447-6267
C.A.C.H.E.

Scott Knaster
5425 E. Kentucky Ave.
Denver, Colorado 80222
(303) 355-2379
Denver Apple Pi

Tony Cerreta
55-A Locust Ave., #4G
New Rochelle, N.Y. 10801
(914) 636-3417
Big Apple Users Group

INTERNATIONAL

7 - AUSTRALIA

Neil Bennett
55 Clarence St.
Sydney, Australia 2000
612-293753
Apple Users Club Australia

9 - NORTH AMERICA

Auby Mandell - Apple Core of Canada
409 Queen St. W.
Toronto, Ontario
Canada M5V2A5
(416) 868-1315

8 - EUROPE

Wolfgang Dederichs
Auf Drenhausen 2
4320 Hattingen
West Germany
02324/67412
Apple Users Group Europe

OFFICERS

March 13, 1980

PRESIDENT:

Ken Silverman
3673 Bassett Ct.
South San Francisco, California 94080
(415) 878-9171
San Francisco Apple Core

VICE-PRESIDENT:

Michael Weinstock
64 Pinedale Rd.
Hauppauge, L. I.
New York 11767
(516) 360-0988
Big Apple Users Group

TREASURER:

Dave Gordon
19273 Kenya St.
Northridge, California 91324
(213) 384-0579
Original Apple Corp.

SECRETARY:

Joe Budge
2507 Elderwood Lane
Burlington, N. C. 27215
(919) 229-6037
Carolina Apple Core

INTERNATIONAL APPLE CORE MINUTES

March 13, 1980

APPENDIX V: TREASURER'S REPORT

RECAP OF ACTIVITIES
THRU 3-9-80

INCOME

DUES	\$ 3,250.00
ORCHARD SALES	30,250.00
ADVERTISING-RECEIVED	18,575.00
ADVERTISING-A/R (MEMO) \$25,675.00	-----
TOTAL RECEIPTS	\$52,075.00

EXPENSES

PRINTING	\$29,730.29
FREIGHT	1,100.00
O/S LABOR, EXPENSES	2,494.54
RENTAL	875.00
REFUND	800.00

TOTAL EXPENSES	\$34,999.83

NET PROFIT (BANK BALANCE 3-9-80)	\$17,075.17

CASH DISBURSEMENTS
THRU 3-9-80

CHECK #	PAYEE	DATE	DESCRIPTION	AMOUNT
1A	Michael Weinstock	12/8	Expenses	\$ 136.00
1B	Carlos Printing	2/2	Printing	10,000.00
2	Ken Silverman	12/14	Exp-P.C.Fund	330.43
	B.S.C.	Jan.		4.20
1001	S. F. Apple Core	1/15	Refund-Orchard	800.00
1002	Matt McIntosh	1/25	Expenses	22.50
1003	Mike Weinstock	1/25	Expenses	171.81
1004	Matt McIntosh	1/25	Expenses	23.05
1005	Carlos Printing	1/29	Printing	9,000.00
1006	Mike Weinstock	2/15	Expenses	371.62
1007	Carlos Printing	2/12	Printing	10,730.29
1008	Computer Faire	2/12	Meeting Room	800.00
1009	Val Golding	2/12	Expenses	43.45
1010	Patricia Boner	2/12	Expenses	45.77
1011	Call Apple	2/12	Expenses	35.00
1012	Patricia Boner	3/1	O/S Labor-Exp.	763.65
1013	Matt McIntosh	3/1	Rental-Advent	75.00
1014	Val Golding	3/4	Freight	545.00
Wire	Val Golding	2/27	Freight	556.50
	B.S.C.	Feb.	Checks	16.34
1015	Matt McIntosh	3/9	Expenses	30.35
1016	Patricia Boner	3/9	Expenses	147.51
1017	Ken Silverman	3/9	Expenses	351.36
TOTAL				\$ 34,999.83

RECAP

Printing	\$29,730.29
Freight	1,100.00
O/S Labor-Expenses	2,494.54
Rental	875.00
Refund	800.00
<hr/>	
TOTAL	\$34,999.83

INTERNATIONAL APPLE CORE MINUTES

March 13, 1980

APPENDIX VI: COMMITTEE CHAIRMEN APPOINTED AT THE GENERAL MEETING

Orchard Publication: Val Golding, 6708 39th Ave. S. W., Seattle, Wash.
98136. (206) 932-6588

IAC Software: Neil Lipson, 29 S. New Ardmore Avd., Broomall, Pa., 19008
(215) 356-6183

New Club Assistance: Dick Sedgewick, 100 Horne St., Dover, N.H., 03820
(603) 742-3703

Constitution and Bylaws: Ken Silverman, c/o International Apple Core
(415) 878-9171

Telecommunications: Craig Vaughn, 3450 E. Spring St., Suite 206, Long Beach,
Calif. 90806 (213) 595-6858 TCA 099

Standards: Mark Robbins, 2726 S. Moline Ct., Aurora, Co. 80014
(303) 755-6440, (303) 696-0200

Newsletter Librarian: Major Terry N. Taylor, 12319 E. Bates Circle,
Aurora, Co., 80014 (303) 750-5813

Newsletter Exchange Coordinator: David Alpert, 880 Melody, Lake Forest,
Ill. (312) 295-6078

Education SIG: Ted Perry, 5848 Riddio, Citrus Hts., Ca. 95610 (916) 961-7776

Handicapped SIG: Bernie Urban, 6205 Walholding Rd., Washington, D. C. 20016
(301) 229-3458

Legal SIG: Butch Clayton, P. O. Box 70278, Charleston Heights, S. C. 29405
(803) 884-5370, (803) 554-9171

INTERNATIONAL APPLE CORE

BOARD OF DIRECTORS

MINUTES OF THE ANNUAL BOARD MEETING

Submitted by Joseph H. Budge - Secretary

March 13, 1980

FIRST SESSION

The first Annual Meeting of the Board of Directors of the International Apple Core was held at 11:00 AM on Thursday, March 13, 1980 at the San Francisco Convention Center in San Francisco, California. This meeting was held during an adjournment of the General Annual Meeting of the International Apple Core (IAC). Mr. Cerreta presided and Mr. Urban took the record. No minutes were read.

QUORUM:

A quorum was determined to be present. Voting members present were Messrs. Allinsky, Wilkinson, Lawrence, Felt, Vitt, Knaster, Urban, Cerreta, Bennett, and Dederichs. No non-voting members were present.

BUSINESS:

After discussion the Board elected the following officers to one-year terms:

President: Ken Silverman
Vice-President: Michael D. Weinstock
Secretary: Joe Budge
Treasurer: Dave Gordon

ADJOURNMENT:

In deference to the General Meeting, the first session of the Annual Meeting of the Board of Directors was adjourned until the General Meeting was finished.

SECOND SESSION

The second session of the Annual Meeting of the Board of Directors of the International Apple Core was held at 3:30 PM on Thursday, March 13, 1980 at the San Francisco Convention Center in San Francisco, California. This meeting immediately followed the first General Annual Meeting of the IAC. The President and Secretary were present as presiding and recording officers, respectively. Minutes of the previous board meeting having been read at the General Meeting, no minutes were read at this meeting.

QUORUM:

A quorum was determined to be present. Voting members present consisted of Messrs. Alinsky, Wilkinson, Lawrence, Felt, Vitt, Knaster, Urban, Cerreta, Bennett, Dederichs, and Silverman. Other persons present but not voting consisted of Messrs. Gordon, Weinstock, Golding, and Budge.

BUSINESS:

Mr. Golding presented an editor's proposal to the Board for future issues of the "Apple Orchard." The Board discussed searching for other editorial candidates, but decided to work with Mr. Golding. This was done to allow continued publication of the "Orchard." Concern was expressed that Mr. Golding would have a conflict of interest between the "Orchard" and "Call-A.P.P.L.E." Mr. Golding stated that, due to the differing editorial contents of the two magazines, he didn't think the problem would occur. The Board decided that all articles would be submitted via the IAC Post Office Box and stamped "For Orchard Only." Specifics of Mr. Golding's proposal were then discussed. The Board shortened the term of the proposal to 18 months, specified publication dates, and tightened the Editor's accounting requirements. As these changes were acceptable to Mr. Golding, the proposal was unanimously approved as amended.

The "Orchard's" editorial content was briefly discussed. Everyone felt that the "Orchard" should be the definitive publication for Apple computers. It was suggested that it be made a journal of only new Apple-related articles. Mr. Golding stated that he intended to include those articles as well as reprint the best of articles from club newsletters and apnotes. This was agreeable to all.

Mr. Golding then proposed that the IAC seek a professional publication firm to put out the "Orchard." He felt that such a firm could do a better printing job, handle subscriptions, advertising, typesetting, and mailing. While this proposal was generally agreeable, the Board expressed concern that insufficient financial information was available to adequately judge the publisher's proposal. Additional concern was expressed that the publishing house could not handle the specialized form of advertising. It was agreed that Mr. Golding would seek a more specific proposal from the publisher while Mr. Weinstock would seek qualified candidates for the position of advertising manager. The prospective advertising manager would be offered no more than 10% of the advertising revenues. The Board agreed to study the proposal further when more information becomes available.

At the request of the international Directors the problem of international representation was discussed. It was felt that two international Directors was an inadequate number. Therefore, the proposed Bylaws were amended to allow one Director from each continent with the exception of Antarctica. The amendment was unanimously adopted. Mr. Mandell was recognized as the Director from the North American Region (Region 9). (See Appendix I).

As discussed during the General Meeting, the international Directors expressed interest in having U. S. Directors represent their interests at future meetings and communicate with them in the interim. This was agreeable to all. Mr. Cerreta agreed to communicate with Europe, Mr. Vitt with Mexico, and Mr. Urban with Canada. Australia would represent itself as Mr. Bennett would be in the U. S. frequently.

Mr. Dederichs showed the Board a catalog his club had made of Apple-related hardware. He suggested that the IAC might wish to distribute the catalog once it was translated to English from German. No action was taken.

ADJOURNMENT:

As it was late, the second session of the meeting was adjourned at 5:10 until the following night.

THIRD SESSION - MARCH 14, 1980

The third session of the Annual Meeting of the Board of Directors of the International Apple Core was held at 7:00 PM on Friday, March 14, 1980 at the Holiday Inn, San Francisco, California. The President was present as presiding officer; Mr. Randy Fields was present in place of the Secretary as recording officer. Minutes of the previous meeting were not read.

QUORUM:

A quorum was determined to be present. It consisted of Messrs. Alinsky, Wilkinson, Lawrence, Felt, Vitt, Knaster, Urban, Cerreta, Bennett, Dederichs, Mandell, and Silverman. Other persons present but not voting consisted of Messrs. Gordon, Weinstock, Lipson, Vaughn, and Fields.

BUSINESS:

It was decided that Mr. Weinstock will call D. C. Hayes to obtain five Micromodemms at cost or for trade in advertising. These will be made available to Directors and Officers who do not own modems. The Modems may be either lent or purchased at cost from the IAC.

After discussion the Board determined that the IAC will make no endorsements for products. If equipment is needed by the IAC it will trade advertising for equipment or services.

An Information Transfer Committee was established. The Committee will consist of the Directors who will get information (Apnotes, etc.) from the Secretary. Directors will then make copies and pass them on to the clubs in their areas. Directors in an area will work out an arrangement between themselves and inform the President of the methods they will use. Directors will receive a \$100 fund to be used for phone calls and mailing of information. The Directors will send in all receipts to the Treasurer before receiving another \$100. Directors are to prepare a budget for submission to the Treasurer next quarter.

The duration of term of the new Directors, one year or two, was decided by the President with the flip of a coin. Messrs. Cerreta, Wilkinson, Felt, and Knaster will serve for two years while Messrs. Alinsky, Urban, Lawrence, and Vitt will serve for one year. All International Directors will serve for two years.

The Board decided to hold the next Annual Meeting in either Boston, Washington, D. C., or Chicago. Directors in the above areas will obtain more information about the local Faires and report back for a decision by the Board in the near future.

It was announced that the IAC represents about 8,000 individuals through its member clubs. All Directors will try to get more clubs to join.

It was determined that the Vice-President will coordinate the writing and sending of a PR release to all magazines.

After discussion it was decided to delay the installation of an 800 number or "Hot Line". Lack of experts and volunteers to answer questions were cited as the primary reasons.

The President asked for, and obtained, permission to add a new phone line to his house for the IAC. It was agreed that Mr. Gordon will obtain Sprint codes and numbers for all the officers and Directors. Billing will go to the Treasurer who will notify the President of abuses.

Mr. Lipson asked about mugs and other specialty items to sell. After discussion it was agreed to make a small amount for a test at a Faire where the IAC would have a booth.

After a long discussion it was motioned and passed to send each member club no more than one diskette each month with software. The diskette will be provided completely free of charge. The production rate will depend on supply: diskettes may not be produced as frequently as once a month unless incoming software is bug free and documented.

It was determined that Mr. Vaughn will assist with standards for documentation.

Use of the Source timesharing network was discussed. The IAC has an account provided at no cost for Officers and Directors. This account may not be used during prime time, as the IAC will be billed for such use. File storage will not be free; again the IAC will be billed. Use of file storage must therefore receive prior approval by the President. Negotiations are proceeding to obtain accounts for member clubs. The IAC will attempt to obtain for member clubs a waiver of the sign on charge (currently \$100). The clubs would still be billed for their own use, however. Mr. Vaughn announced that Peripherals Unlimited will give all Directors and Officers a copy of the Source sign-on program in exchange for the \$200 sponsoring fee. This was agreed to by the Board.

Mr. Weinstock announced that he had obtained fifteen free accounts for IAC use on Micronet. These accounts are also for Directors and Officers. They will be used for local club type mail in both directions. Mr. Weinstock will send all information to the Directors and Officers.

The Directors determined to work out a plan for voting in their region by member clubs. That plan is to be written up and given to the IAC Secretary. This will allow new clubs joining in an area to learn how to vote and how to contact their directors. International as well as national Directors will prepare such plans.

It was determined that the IAC will look into assembling an Apple training program. Mr. Urban and Mr. Fields will work on the project. This will take a great deal of time. It is hoped to have a concrete outline and plan in time for the next Annual Meeting.

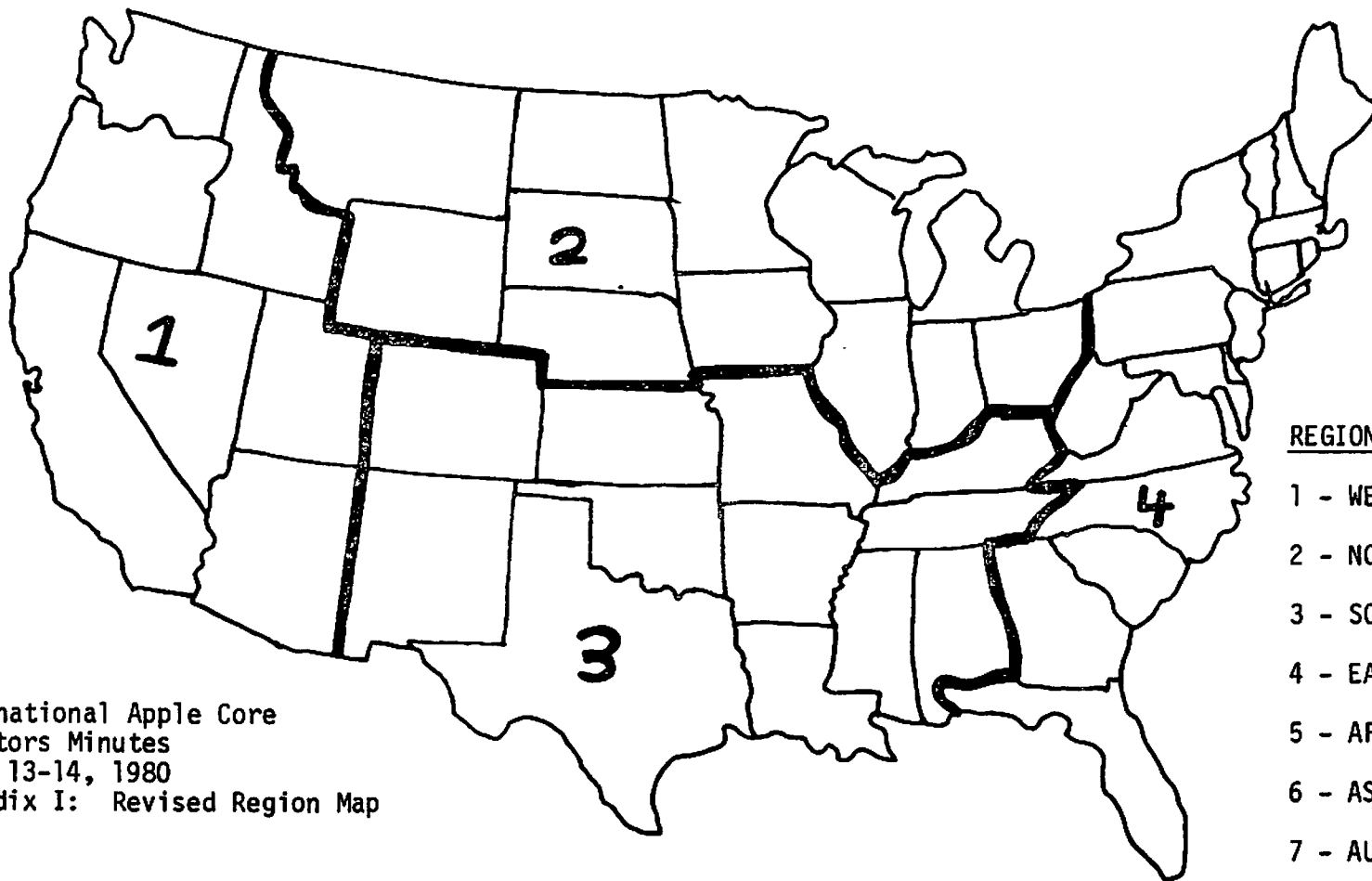
After discussion it was determined that outgoing Directors will not be paid to attend an Annual Meeting. When a new Director is voted into the Corporation the Secretary is to be notified as soon as possible.

Full Member dues per year were set at \$50.00 independent of membership size. This is a nominal fee as it doesn't even cover the price of the monthly diskettes. The dues will be payable in January 1981. If a club joins this year it will pay only a \$50.00 initiation fee. Next year it will pay both the initiation fee and the dues for that year, a total of \$100.

Sponsoring membership dues were set at \$200.00 per year. Membership, once paid, will be valid for a full 12 months from the date of payment.

ADJOURNMENT:

The first Annual Meeting of the Board of Directors of the International Apple Core was adjourned at 10:00 PM.



REGIONS:

- 1 - WEST
- 2 - NORTH
- 3 - SOUTH
- 4 - EAST
- 5 - AFRICA
- 6 - ASIA
- 7 - AUSTRALIA
- 8 - EUROPE
- 9 - N. AMERICA
(ex. U.S.A.)
- 10 - S. AMERICA

International Apple Core
Directors Minutes
March 13-14, 1980
Appendix I: Revised Region Map

INTERNATIONAL APPLE CORE

BOARD OF DIRECTORS

MINUTES OF A SPECIAL MEETING

Submitted by Joseph H. Budge - Secretary

May 19, 1980

A special meeting of the Board of Directors of the International Apple Core (IAC) was held at 4:35 PM on Monday, May 19, 1980 in the conference room of the Sunshine Restaurant in Anaheim, California. The President and Secretary were present as presiding and recording officers, respectively. Synopses of the minutes from the previous Board meetings were read. Approval was deferred.

QUORUM:

A quorum was determined to be present. Voting members present consisted of Messrs. Alinsky, Urban, Wilkinson, Felt, Vitt, Mandell, Knaster, and Silverman. Those present but not voting consisted of Messrs. Gordon, Weinstock, Vaughn, Golding, and Budge.

REPORTS OF OFFICERS:

The President reported that organization of the IAC was proceeding according to plan.

The Vice President reported that tasks identified for him in the previous Board meeting had been accomplished. Five D. C. Hayes Micromodems had been obtained in exchange for advertising. Three had been resold to Officers or Directors. Micronet hookups had been finalized and obtained for the Directors and Officers.

The Secretary reported that the IAC files and records had been organized, that three mass mailings had been produced, and that club correspondence had been handled.

The Treasurer reported that the IAC held \$26,000 in net cash, \$10,000 of this was in a time deposit savings account which had been opened. Receivables consisted of \$8,800 of unpaid Orchard ads and \$4,000 of unpaid Orchard sales. 19,000 Orchards remained unsold, representing another potential \$19,000 in future revenues. The Treasurer reported that he had tried to determine a budget for the IAC but had dropped the attempt because no historical data were available yet. The President proposed that, until a full-time bookkeeper was hired, the Treasurer's report be produced quarterly instead of monthly. The motion was unanimously approved.

REPORTS OF DIRECTORS:

Mr. Urban reported that he had written every club in the Eastern area in attempting to organize an area caucus. In a month he had received no responses, merely some calls with comments on IAC service.

Mr. Wilkinson reported that he had telephoned most clubs in his portion of Area 1. No response had been generated towards developing an area caucus

because no one had had time to think about it. He also reported that the mailings had been onerous.

Mr. Alinsky agreed with the previous reports. He found that clubs came to him mostly with questions about what the IAC should be doing.

Mr. Vitt reported that he had visited with 17 clubs in his area. He had found personal contact with the clubs very helpful.

Mr. Felt reported that he had contacted clubs in his area and attended many meetings. He requested that Directors be reimbursed for expenses incurred in travelling to club meetings. In addition he reported that he had talked with many dealers in his region and had sent each a letter requesting support for the IAC. He suggested that each director send a similar letter in each area. The Secretary will provide a copy to all Directors.

BUSINESS:

A postage machine was proposed for use in mailings. After discussion the proposal was dropped as impractical and expensive.

A question was raised over the propriety of Directors or Officers conducting business with the IAC. After discussion it was decided that no current action was necessary, as no abuses were evident. All motions on the floor were removed. It was understood that the Treasurer's regular report would provide the proper checks on potential abuses. It was further understood, but not mandated, that all officers would be consulted prior to possible conflicts of interest.

A motion was made that any outgoing Directors, not re-elected by their regions be flown to the Annual Meetings as well as the Directors - elect. The motion was seconded and passed, reversing the decision made at the previous meeting of the Board.

In view of the Directors' lack of success in organizing area caucuses, it was proposed that the Board include election procedures in the Bylaws. After discussion a committee of two-year Directors was appointed. The committee will report their recommended Bylaws additions by June 30. The proposal and committee were approved with no negative votes and one abstention.

Concern was expressed over the large number of "Orchards" which remained unsold. It was reported that solicitation letters to dealers had just been mailed, so inadequate time had passed for a response. It was suggested that if clubs found that they had too many copies on hand the copies could probably be sold to local dealers. No action was taken.

The President showed samples of airbrushed IAC Tee-shirts which had been mentioned at the previous Board meeting. After pointing out that their quality was inadequate under the IAC's pending trademark agreement with Apple Computer, Inc., adoption of these Tee-shirts was unanimously vetoed.

A motion was made that the President and Treasurer be empowered to sign legal documents on behalf of the IAC. After discussion the motion was amended to: "All officers are empowered to enter into contractual agreements on behalf of the International Apple Core." It was stipulated that the resolution was subject to final and appropriate wording by legal counsel. The motion as so amended and stipulated was passed by a 6-1 vote.

The President was specifically authorized to sign checks for the IAC in the event that the Treasurer could not.

The time and place for the next annual meeting was discussed. As persons delegated to study the Boston and Chicago sites were not present at the meeting, their reports were not available. After discussion the Board dropped Philadelphia and Washington as possible sites and favored Boston. It was determined

that Messrs. Urban and Cerretta will coordinate investigation of the Boston site and Messrs. Felt and Lawrence will do the same for the Chicago site. The President directed that the chosen site must have a club and person responsible for organizing the IAC meetings and booths. The meetings will be held for two days prior to the associated convention; one day for the General Meeting and one day for the Director's Meeting.

Mail and communications standards for the IAC were discussed. The Secretary will provide the necessary letterhead and envelopes.

A motion was made that all mass mailings to clubs be made through a central service. The service would alleviate Directors from the burdens of mass mailings and would insure that all clubs would receive mailings more or less simultaneously. The service would also mail software. After discussion the motion was approved. The Board directed the secretary to pre-announce all mailings and their contents via telecommunications. The Directors and Officers would then have the opportunity to draft cover letters and transmit them to the Secretary for inclusion in mailings to their constituency.

A motion was made that the IAC make available extra copies of Apnotes and other appropriate materials in bulk and at cost to member clubs. It was felt that clubs could make their own copies. The motion was unanimously defeated.

The Secretary was directed to obtain from all clubs the times and places of their regular meetings so that the Directors may attend.

Grawin Publications proposal for publication and advertising management of the "Orchard" was put on the floor for discussion. Concern was expressed on two points: First that Grawin's advertising proposal appeared unrealistic, and secondly that no alternative proposals were available for competitive evaluation. After discussion it was moved, seconded, and passed that the IAC contract with Grawin to publish and organize advertising for no more than three issues of the "Orchard" while a committee investigated alternatives. Mr. Alinsky and Mr. Weinstock were appointed to the Publication Committee. The Committee was charged to look at publishers, advertising managers, and all aspects of publication. The Committee is to provide cost and revenue analyses of alternative methods of publication and of generating advertising. If a publisher other than Grawin is eventually selected, suitable arrangements for communications between publisher and editor must be feasible. The preliminary report of the Publication Committee must be submitted by the time of publication of the next Orchard (September 1, 1980). Mr. Golding recommended that, for logistical purposes, a decision on a future publication agreement should be made by the time that the third issue of the "Orchard" goes to press.

It was motioned that the Secretary be provided a tie line for local linkage to the telecommunications networks. After discussion this was deemed to be an operating decision best left to the Officers.

ADJOURNMENT:

The special meeting of the Board of Directors of the International Apple Core was adjourned at 10:00 PM.



INTERNATIONAL APPLE CORE

April 15, 1980

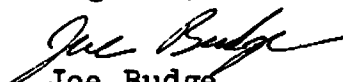
Members:

The International Apple Core is pleased to enclose its first issue of application notes. You will find:

- A list of known bugs in Pascal
- A fixed quick input routine
- A fix for the Append command in DOS 3.2
- A list of the graphics entry points in Applesoft
- A keyboard modification that allows some extra characters

I recommend that you prepare a three ring binder to hold these and future application notes. Many more Apnotes will be forthcoming. In addition to new ones, Apple Computer has forwarded their entire package of Apnotes. Clubs joining the IAC prior to 3/13/80 recieved this package in rough form. We will be cleaning them up and republishing them as time permits. I currently am working to classify the Apnotes in a sensible fashion.

Regards,


Joe Budge,
Secretary



INTERNATIONAL APPLE CORE

APNOTES

From time to time the International Apple Core will distribute application notes to our members. Application notes are technical corrections or improvements to Apple Computers or their software. Usually application notes address themselves to specific problems. Since other users share the same problems, we will make every effort to publish them on a timely basis.

The International Apple Core solicits interested users and manufacturers to submit application notes for publication. Appropriate subjects might be:

- Hardware modifications to fix a design error
- Hardware modifications to upgrade a system
- Software patches
- Software modifications to upgrade a program
- Modifications to existing software/hardware that add new features to the Apple Computer system.
- Notification of bugs and the conditions under which they occur.
- Documentation to previously undocumented hardware or software

Application notes are not restricted to Apple Computer Company's products. They may apply to anyone's, provided they will be of interest to Apple users.

To expedite application note publication, we ask you to submit notes in publishable form. This is not a requirement. We feel it is more important to have an Apnote in any shape than to have a spiffy document. If you lack time or facilities the IAC will prepare final drafts for you. Publishable form means that text will be typed on white paper, preferably with a dark ribbon. Single space and use a readable character size. Page margins are 7.25 inches by 9.0 inches. Drawings should be neat and legible. The closer you can approximate this, the faster the Apnote can be published. Purley textual Apnotes can also be submitted in text files on diskette. Mail them to:

Apnotes
International Apple Core
P.O. Box 976
Daly City, Calif. 94017

If you wish to assist in the review and "cleaning up" of Apnotes send us a note or call the IAC Secretary.

P.O. BOX 976 DALY CITY, CALIFORNIA 94017



Application Note

P.O. BOX 976 DALY CITY, CALIFORNIA 94017

STALKING THE WILD "€"

OF THE 96 STANDARD ASCII UPPER CASE AND CONTROL CHARACTERS, THE APPLE KEYBOARD CAN GENERATE ALL BUT 5. NAMELY, "€" (\$DB), "\" (\$DC), "_" (\$DF), AND CONTROL CHARACTERS F5 (\$9C) AND US (\$9F). WHO CARES? WELL, CERTAIN FANATICS CAN'T STAND FOR THEIR APPLES TO BE LESS THAN PERFECT. AT EVERY OPPORTUNITY THEY "IMPROVE" THEIR MACHINES BY ADDING LOWER CASE ADAPTERS AND SHELLING OUT 5 BIG ONES FOR THAT LAST 16K RAM. NOTHING CAN STAND IN THE WAY OF THEIR QUEST FOR THE ULTIMATE APPLE.

THE KEYBOARD SHORTCOMING HAS LONG BEEN A THORN IN THE SIDE OF THESE APPLE ADDICTS AND VARIOUS CRUTCHES HAVE BEEN DEVELOPED. APPLESOFT CHR\$ ALLOWS PRINTING OF THESE CHARACTERS—TACKY AND INCOMPLETE. ONE CAN TRAP AND MODIFY VARIOUS CONTROL CHARACTERS IN A SPECIAL KEYBOARD INPUT ROUTINE, BUT THEN THE CONTROL CHARACTERS JUST TRAPPED CAN'T BE USED—NOT ELEGANT. THE SOLUTION INVOLVES HARDWARE MODIFICATIONS. SO CRANK UP THOSE SOLDERING IRONS OUT THERE KIDDIES—HERE IT COMES.

THE L, K, AND O KEYS ON APPLE KEYBOARD ARE MERELY CONNECTED ACROSS THE WRONG MATRIX PINS ON THE NATIONAL SEMI MM5740 KEYBOARD ENCODER (NOT REALLY WRONG, JUST NOT THE BEST CHOICE, PERHAPS, SEE FIGURE 1). THESE KEYS ARE CURRENTLY CONNECTED AS FOLLOWS.

K CONNECTS X4 TO Y9,
L CONNECTS X4 TO Y8,
O CONNECTS X3 TO Y9.

THE IMPROVEMENT IS AS FOLLOWS.

K CONNECTS X4 TO Y4,
L CONNECTS X4 TO Y5,
O CONNECTS X4 TO Y3.

WITH THIS STRAIGHT FORWARD CHANGE, SHIFT-K GENERATES "€", SHIFT-L GENERATES "\" , SHIFT-O GENERATES "_" , CONTROL-SHIFT-L GENERATES F5, AND CONTROL-SHIFT-O GENERATES US (CONTROL-SHIFT-K GENERATES ESC).

THE MODIFICATION IS DELICATE AND VOIDS THE WARRANTY, BUT IS GUARANTEED TO PROVIDE GREAT RELIEF FOR MANY HARDWARE FREAKS CRAVING SUCH A FIX. INSTRUMENTS REQUIRED FOR THE OPERATION ARE SOLDERING IRON, SOLDER, 4 SHORT LENGTHS OF HOOKUP OR WIRE-WRAP WIRE, SOLDER-WICK OR SOLDER-SUCKER, X-ACTO KNIFE, #2 PHILLIPS SCREWDRIER, STEADY HANDS, AND PATIENCE.

FIRST REMOVE KEYBOARD FROM APPLE. THE K, L, AND O KEYS MUST NOW BE REMOVED FROM THE APPLE TO ALLOW ACCESS TO CIRCUIT TRACES BENEATH THEM. CAREFULLY REMOVE ALL SOLDER FROM THE KEYSWITCH TERMINALS AS THEY ARE FRAGILE AND MUST NOT BE FORCED OUT. CUT THREE TRACES AS SHOWN IN FIGURE 2. SOLDER SWITCHES BACK IN PLACE. CUT ONE TRACE ON BACK OF KEYBOARD AND ADD 4 JUMPERS AS SHOWN IN FIGURE 3. IF YOUR KEYBOARD TRACES DON'T FOLLOW THE PATTERN OF FIGURES 2 AND 3 (NEWER VERSIONS DO NOT), VERIFY CONTINUITY FROM SWITCH TERMINALS DIRECTLY TO THE X4, Y3, Y4, AND Y5 PINS ON THE KEYBOARD ENCODER. REASSEMBLE APPLE AND PUT YOUR TOOLS AWAY. THAT'S IT!

YOU MAY NEVER NEED YOUR NEW CHARACTERS, BUT DON'T YOU FEEL MORE SECURE JUST KNOWING THEY'RE THERE.

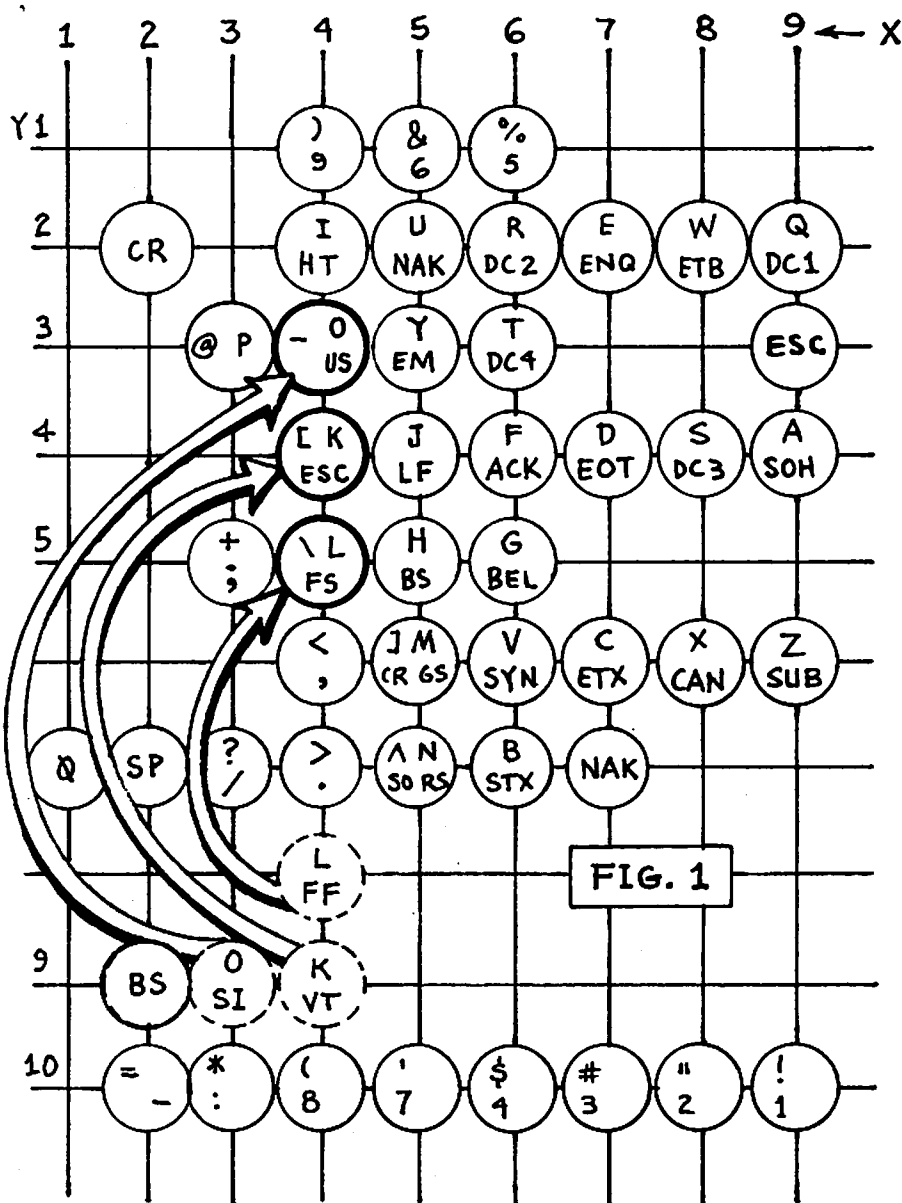
FIGURE 1. SCHEMATIC OF APPLE KEYBOARD SHOWING MODIFICATION TO PRINT "€", "\" , "_" , AND " " . (ADAPTED FROM APPLE II REFERENCE MANUAL, PAGE 181.)

FIGURE 2. VIEW UNDER K, L, AND O KEYS.

FIGURE 3. VIEW OF BACK SIDE OF KEYBOARD.

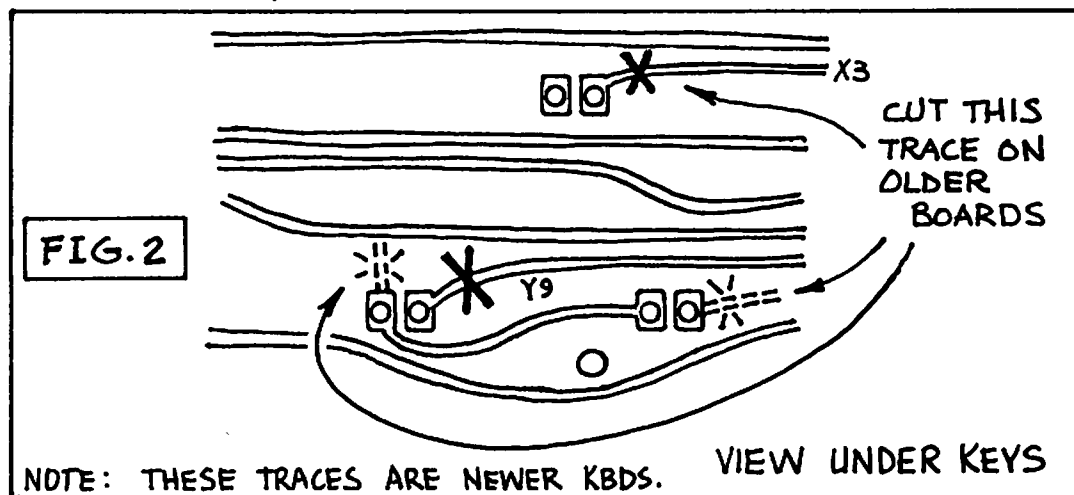
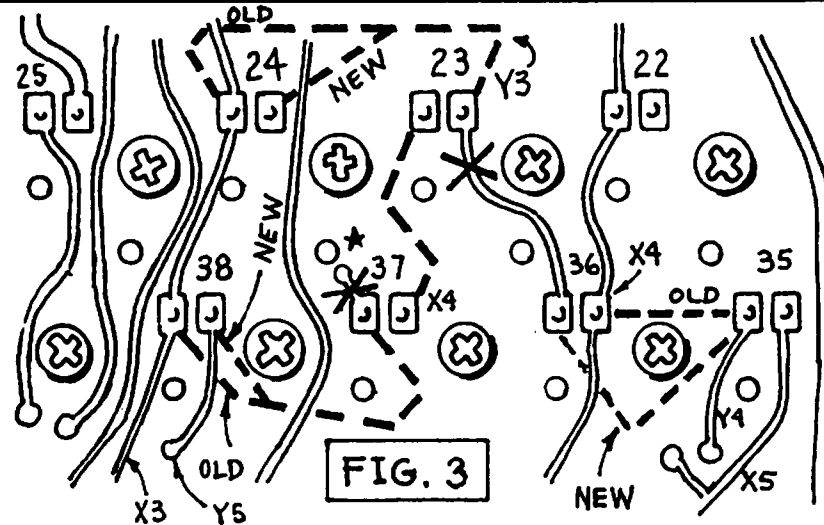
DAVID HUDSON
15 DECEMBER 1979

A6-1



SEE P. 101 OF APPLE II REFERENCE MANUAL

This application note has been provided by an Apple Computer user. The International Apple Core does not guarantee the accuracy of this information in any way and cautions that modifications may void a manufacturer's warranty. Apple II, Applesoft, and Apple Computer are trademarks of Apple Computer Company, Inc.



NOTE: THESE TRACES ARE NEWER KBDS.

ASSY. NO 01-0425-02

PC. 02-2459-02 REV.A

CAUTION: IF YOU HOOK THIS UP WRONG AND APPLY POWER, YOU CAN KISS YOUR KEYBOARD ENCODER GOODBYE.

Application Note



KEYBOARD MODIFICATION

P.O. BOX 976 DALY CITY, CALIFORNIA 94017



Application Note

P.O. BOX 976 DALY CITY, CALIFORNIA 94017

APPLESOFT HIRES ROUTINES

These are the entry points to program machine level high resolution graphics with the routines in ROM APPLESOFT.

ZERO PAGE LOCATIONS

March 12, 1980

HPAG \$E6 \$20=PAGE 1,\$40=PAGE 2
ROT \$F9 USED WITH SHAPE ROUTINES
SCALE \$E7 USED WITH SHAPE ROUTINES

These entry points supersede those published in the March, 1980 "Apple Orchard", V1N1p17.

B2

THE GRAPHICS ROUTINES

HGR2 \$F3D8
INITIALIZE TO PAGE 2

HGR \$F3E2
INITIALIZE TO PAGE 1

HCLR \$F3F2
CLEAR CURRENT SCREEN TO BLACK

BKGND \$F3F6
CLEAR CURRENT SCREEN TO LAST PLOTTED HCOLOR

HPOSN \$F411
POSITIONS HIRES CURSOR WITHOUT PLOTTING. A=Y X=XL Y=XH

HPLOT \$F457
POSITIONS HIRES CURSOR AND PLOTS A POINT. A=Y X=XL Y=XH

HLIN \$F53A
DRAWS A LINE FROM LAST POINT TO A=XL X=XH Y=Y

HFIND \$F5CB
CONVERT LAST POINT'S POSITION TO X-Y COORDINATES.
THE RESULTS ARE IN \$E0=XL \$E1=XH \$E2=Y

DRAW \$F601
SHAPE DRAW SUB. A=ROTATION FACTOR, X=SHAPE TABLE LOW,
Y=SHAPE TABLE HIGH. USES SCALE.

XDRAW \$F65D
SAME AS DRAW EXECPT DOES AN EXCLUSIVE OR WHEN PLOTS ON THE
SCREEN.

SETHCOL \$F6EC
X=COLOR. MUST BE LESS THAN 8.

SHLOAD \$F775
LOADS SHAPE TABLE ABOVE HIMEM (\$73,74)



Application Note

P.O. BOX 976 DALY CITY, CALIFORNIA 94017

March 10, 1980

LITERAL INPUT

Here is another garbage collection forestaller with some nice additional benefits. It allows you to enter commas, quotes, and colons into Applesoft without getting an "EXTRA IGNORED" error for your efforts and works just as well for either keyboard or disk input. Originally published in CONTACT 6, the routine has been revised to remove some errors. Here's what it does:

- Line 100 defines a string variable at a known memory location. (This name can be anything you wish. We just happen to use IN\$.)
- Lines 220-290 poke a short machine language routine into page 3 of your Apple's memory. This routine changes the pointer to the string in memory to point at the input buffer (\$200).
- Line 350 calls the new input routine and the MID\$ function moves a copy of the new string into main memory so that it isn't overwritten by the next input.

Here's a cute trick for using this routine with random access disk files. Say your program is reading a file for the third field in each record. Using this routine, the syntax for that would be:

```
CALL 768: CALL 768: CALL 768: IN$=MID$(IN$,1)
```

The first two calls are dummy INPUTS but, unlike the normal DOS "INPUT IN\$" command, perform no string operations.

Try it. You'll like it.

B6-1

```

100 LET IN$ = "X"
110 TEXT : HOME
120 REM
130 REM THE FIRST VARIABLE
140 REM DEFINED MUST BE A STRING
150 REM THIS STRING WILL REC'VE
160 REM INPUT FROM THE CALL
170 REM
180 REM THIS POKES THE INPUT
190 REM SIMULATOR ROUTINE
200 REM INTO MEMORY...
210 REM
220 FOR J = 768 TO 790
230 READ I
240 POKE J,I
250 NEXT J
260 DATA 162,0,32,117,253,160,2
270 DATA 138,145,105,200,169,0
280 DATA 145,105,200,169,2,145
290 DATA 105,76,57,213
300 REM
310 REM NOW TO USE IT!
320 REM
330 PRINT "TYPE IN ANY CHARACTERS YOU WISH:"
340 PRINT
350 CALL 768:IN$ = MID$ (IN$,1)
360 REM
370 REM THIS IS AN "INPUT IN$"
380 REM BUT IGNORES ",", "&" & ":"
390 REM
400 PRINT
410 PRINT "AND HERE'S WHAT YOU TYPED IN:"
420 PRINT : PRINT IN$
430 PRINT
440 PRINT "NOTE THAT EVEN QUOTES, COMMAS AND"
450 PRINT "COLONS GET THROUGH UNSCATHED."
460 PRINT : PRINT "NOW LET'S WRITE IT TO THE DISK."
470 PRINT CHR$ (4)"OPEN TEMP"
480 PRINT CHR$ (4)"WRITE TEMP"
490 PRINT IN$
500 PRINT CHR$ (4)"CLOSE"
510 PRINT : PRINT "AND READ IT BACK IN..."
520 LET IN$ = " "
530 PRINT CHR$ (4)"OPEN TEMP"
540 PRINT CHR$ (4)"READ TEMP"
550 CALL 768:IN$ = MID$ (IN$,1)
560 PRINT CHR$ (4)"CLOSE"
570 PRINT : PRINT IN$
580 PRINT : PRINT "TA-DAA!!": END

```

34. Use of the compiler swapping option causes global declarations to be ignored.
35. The manual does not clearly state that the string argument of the .TITLE pseudo-operation must be enclosed in double quotation marks.
36. Not all options of the Editor are listed on the prompt line, and the "?" option to get a list of the additional options is not supported.
37. The reference manual does not adequately describe how to set up a program to be executed automatically, using SYSTEM.STARTUP.
38. The Editor only accepts files with names of the form <file>.TEXT.
39. The Editor is insensitive to certain punctuation conventions during paragraph reformatting.
40. The Assembler should translate all alphabetics to upper-case.
41. The documentation is inadequate in the description of assembly language and various options.
42. The Compiler may generate an error #407 (too many libraries).
43. The compiler generates code to always load segments 28-31 if they are present. This may cause undesired loading.
44. When the NOLOAD option is used for an intrinsic unit with a data segment, the compiler fails to generate code to unload the data segment at the end of execution. Thus, if the program is run again, the data segment will not be loaded since the interpreter segment table will show that it is already there.
45. The ATAN function gives an erroneous result for an argument less than -1.00.
46. BIOS does not turn off the high-order bit of characters handled by the remote I/O routines.
47. The P1 parallel printer PROM appears to be incompatible with BIOS. The P1-02 PROM is required.
48. The listing file produced by the Assembler does not follow the standard for text files as described in the Apple PASCAL Manual.

49. There is some confusion about the differences between the built-in variable KEYBOARD and the volume identifier SYSTEM:.
50. The fact that the Editor requires tedt files to have an even number of 512-byte blocks is not documented.
51. If a run-time error occurs while in graphics mode, the screen does not switch back to text mode to display the error message.
52. The Cross Reference program on APPLE3 does not close and lock the output file.
53. The Compiler does not check the declared length of STRING parameters passed by reference. Therefore, if the declared length of the actual parameter is less than that of the formal parameter, assignment of characters into the formal can clobber space beyond the end of the actual parameter without detecting any error condition.
54. Using an uninitialized long integer may cause unpredictable problems because there may be illegal bit patterns that are not representations of digits.
55. The Compiler QUIET option (Q+) does not turn off all output to the console.
56. The system will crash if the system disk containing the Editor is not on line when returning from a Copy File command when the system disk was replaced for a disk containing the file to be copied.
57. The interpreter code for floating point comparisons returns $0.0 > (-0.0)$.
58. DIV and MOD functions give incorrect results for certain combinations of signs for the arguments.
59. The code to clear SYSCOM is incorrect in the procedure BOOT.
60. The figures given for the maximum integer values for each declared length of long integer are incorrect in the Apple PASCAL Manual, page 198.
61. Source code for APPLE3:LINEFEED should be made available.



Application Note

P.O. BOX 976 DALY CITY, CALIFORNIA 94017

March 10, 1980

This is a list of known PASCAL problems. Updates and fixes will be announced

1. The integer value -32768 prints as "--2768" and causes a compile error in the expression I:=-32768 where I:Integer.
2. A long integer compare causes the system to crash.
3. The compiler does not allow R:Real; I,J:Integer; R:=I/J; which should be legal according to Jensen & Wirth page 147.
4. The BREAK key (ctrl-shift-P) does not cause a break during the execution of some programs.
5. A variable of type TEXT can be passed as a VAR parameter. This is OK according to Jensen & Wirth, page 157.
6. The Editor sometimes ends a file with 00 instead of the required OD00. This results in trash on the screen.
7. TTLOUT in Applestuff does not work.
8. Transcendental functions are not included in the APPLE3:CALC program.
9. The compiler will allow more than 9 segment procedures but only 9 will function properly.
10. Separate units do not work.
11. Erroneous placement of control characters may cause the Editor to go out of control, requiring the user to delete text in order to recover.
12. When compiling using the (*\$L+*) option, the compiler may damage the contents of the diskette.
13. Intrinsic units cannot use non-intrinsic units, and vice versa.
14. Intrinsic units cannot contain references to files.
15. Long integer constants are not implemented.
16. The IN function for set inclusion does not always work when the first argument is negative or greater than 511.
17. A run-time stack overflow crashes the system instead of re-initializing.

This application note has been provided by an Apple Computer user. The International Apple Core does not guarantee the accuracy of this information in any way and cautions that modifications may void a manufacturer's warranty. Apple II, Applesoft, and Apple Computer are trademarks of Apple Computer Company, Inc.

Pascal (G) 13,p.1

18. The MEMAVAIL function may return incorrect results in some cases.
19. .PUBLIC and .PRIVATE assembler variables may be relocated incorrectly at run-time.
20. There is no way to tell the compiler to allocate all available disk space for the code file, so the compiler may run out of room for the code file even if sufficient space is available.
21. The ORD function accepts REAL and pointer arguments even though this is incorrect.
22. Negation of BOOLEAN variables do not turn FALSE to TRUE and vice versa.
23. The compiler allows underscores in an identifier, but ignores them.
24. Functions cannot return STRING values, although this is implied in the Apple PASCAL Reference Manual.
25. Standard PASCAL syntax allows the field list and <variant> in a RECORD declaration to be null. The UCSD compiler does not allow these items to be empty.
26. Due to the normal inaccuracies in representing REALs, some equalities may not test true. For example, LOG(10) prints as 1.00000 but does not yield equality in the comparison LOG(10)=1.00000.
27. Documentation for the SCAN function is incomplete.
28. The example of MOVERIGHT is incorrect.
29. The Editor does not report assembler errors in the same way that it reports compiler errors.
30. The function KEYPRESS exists in Applestuff, but is not documented.
31. Overflowing the code file causes the system to crash in various ways.
32. The Editor informs the user when it is about to run out of space for the file buffer, but strange things may happen when the file is completely full.
33. The compiler INCLUDE directive does not always work as expected.



Application Note

P.O. BOX 976 DALY CITY, CALIFORNIA 94017

March 15, 1980

J2

APPEND FIX IN DOS 3.2 (& 3.2.1) FROM APPLE COMPUTER CORP.

The problem with APPEND in DOS 3.2 is that DOS doesn't write an End Of File marker on the disk when you chose a file. DOS normally fills new sectors with EOF markers, so the newly APPENDED information usually has an EOF after the last character. However, when the last character of the file falls exactly at the end of a sector, DOS doesn't find a new sector to fill with EOF markers. The next time DOS does an APPEND it can't find the EOF marker and defaults back to the beginning of the file.

The fix is to write out an EOF maker before closing the file after each write. Here is a five byte routine that will supply an EOF. It can be moved to any address if you are already using 768 to 772.

```
10 LET D$= CHR$(4)
20 POKE 768,169
30 POKE 769,0
40 POKE 770,32
50 POKE 771,237
60 POKE 772,253
70 REM HOW TO USE IT--
80 PRINT D$: "APPEND FILE"
90 PRINT D$: "WRITE FILE"
100 PRINT "THIS IS DATA"
110 PRINT "SO IS THIS"
120 CALL 768: PRINT: REM THI IS IT
130 PRINT D$: "CLOSE FILE"140 END
```

NOTE: The PRINT statement in line 120 is a must.

Using this method, one need never worry about APPEND overwriting the start of a file.



June 10, 1980

Dear Member Clubs:

The enclosed package represents the International Apple Core's first major publication of Apnotes. You will find a mix of new and old notes. The older ones are reprints based on poor-quality copies which have been distributed in the past from various sources. This printing covers about one-half of the IAC library of reprints. The remainder will be published as they are printable. While reprinting old Apnotes, we will continue publishing new ones on as timely a basis as is possible.

You may have noticed that each Apnote carries a letter and number code. These correspond to the codes which the IAC uses for indexing. An index for Apnotes published to date is attached.

There has been considerable confusion over the IAC address. In addition to having our own address each of our officers, directors, and committee members has a home address as well. No one knows where to send the mail! In general we prefer to have all mail sent to the IAC Post Office Box. This includes articles or other correspondence for the "Orchard". An efficient distribution system has been set up to see that the mail gets quickly routed to the proper individual. This way personnel changes and moves won't disrupt our timely response.

A new IAC Special Interest Group is being formed. Larry L. Stoneburner, M.D. will organize the Medical SIG. Interested persons can contact Dr. Stoneburner by writing the IAC or at 2030 E. 4th Street, #133, Santa Anna, CA. 92705. His telephone number is 714-953-9151.

Regards,

Joe Budge
Secretary



APNOTE INDEX

October 15, 1980

A. HARDWARE MODIFICATIONS

- 2. User Firmware (2716)
- 4. Adding Colors to Hires
- 6. Five Additional characters from the Keyboard

B. BASICS

- 2. Applesoft Hires Routines
- 3. Applesoft Array Eraser
- 4. Applesoft Hires Screen Function
- 5. Generating Tones in Applesoft
- 6. Literal Input Routine
- 8. Print Using Simulator
- 9. Converting Integer Basic Programs to Applesoft
- 10. Applesoft Random Numbers
- 11. Applesoft Out of Memory
- 12. VTAB and HOME Converter for Sup-R-Terminal

C. MACHINE LANGUAGE

- 8. Adding Features to LISA
- 9. Putting Programma M/L onto Disk

D. INTERFACING INFORMATION

- 2. Cassette Interface
- 4. DEL Character Killer
- 5. Correction To Sup-R-Terminal Preliminary Manual

E. PRINTER INTERFACING

- 3. Serial Handshake Modification with Tabs

F. DATASHEETS

- 5. Apple Post

G. PASCAL

1. Lower Case Patch
2. Linefeed
3. Take 280
4. Getrem
5. Transfer & Sum 512
7. Foreign & Gettext
9. Comm Card Baud Rate Changer
10. Interfacing Foreign Hardware
11. Long Integer Fix
12. Hires
13. Known Pascal Bugs
16. Pascal Units
17. Pascal Peeks & Pokes

H. TEXT AND GRAPHICS INFORMATION

1. Text Screen Mapping and Use

I. LISTS

J. DOS

1. DOS Demo Programs
2. 3.2.1 Append Fix



INTERNATIONAL APPLE CORE

APNOTE INDEX

JUNE 10, 1980

- A. HARDWARE MODIFICATIONS
 - 2. User Firmware (2716)
 - 4. Adding Colors to Hires
 - 6. Five Additional characters from the Keyboard
- B. BASICS
 - 2. Applesoft Hires Routines
 - 3. Applesoft Array Eraser
 - 4. Applesoft Hires Screen Function
 - 5. Generating Tones in Applesoft
 - 6. Literal Input Routine
 - 8. Print Using Simulator
 - 9. Converting Integer Basic Programs to Applesoft
 - 10. Applesoft Random Numbers
 - 11. Applesoft Out of Memory
 - 12. VTAB and HOME Converter for Sup-R-Terminal
- C. MACHINE LANGUAGE
 - 8. Adding Features to LISA
 - 9. Putting Programma M/L onto Disk
- D. INTERFACING INFORMATION
- E. PRINTER INTERFACING
- F. DATASHEETS
- G. PASCAL
 - 2. Linefeed
 - 3. Take 280
 - 4. Getrem
 - 5. Transfer & Sum 512
 - 7. Foreign & Gettext
 - 9. Comm Card Baud Rate Changer
 - 11. Long Integer Fix
 - 12. Hires
 - 13. Known Pascal Bugs
 - 16. Pascal Units
- H. TEXT AND GRAPHICS INFORMATION
- I. LISTS
- J. DOS
 - 1. DOS Demo Programs
 - 2. 3.2.1 Append Fix



MODIFYING YOUR APPLE II TO ACCEPT USER FIRMWARE

There are times when you may want to create your own firmware for the Apple II. Such firmware can be tailored to your special needs; then installed semi-permanently so that it can be used as conveniently as Apple BASIC or the Monitor.

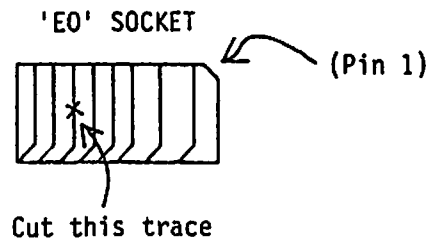
This application brief details a simple modification which allows your Apple II to accept industry standard 2716 (2Kx8-bit) Erasable Read-Only-Memories (EROM's) in sockets 'D0' and 'D8'. These sockets correspond to memory addresses D000-DFFF.

2716 EROM's are readily available through most semiconductor distributors. Many distributor locations are equipped to program the EROM's to your specifications, and will do so for a moderate fee.

NOTE: THE ABOVE INFORMATION IS PROVIDED FOR THE CONVENIENCE OF THE SOPHISTICATED OWNER. ANY USER MODIFICATION TO THE APPLE II (INCLUDING THIS ONE) VOIDS THE FACTORY WARRANTY.

APPLE-II 2716 EROM ADAPTATION INSTRUCTIONS
('DO' and 'D8' sockets)

1. Remove the 'EO' ROM from its socket. On the top side of the board, under the 'EO' socket, cut the ROM pin 18 jumper trace. Then reinsert the ROM. This cut will isolate pin 18 of ROM 'DO' and 'D8' from pin 18 of the other ROM. Reinsert the 'EO' ROM when done.



2. On the underside of the APPLE-II board, cut the traces connecting pin 20 to 21 of ROMs 'DO' and 'D8' only.
3. (Underside) Cut the trace going to pin 18 of ROM 'D8' near the chip. Scrape solder resist off of approximately $\frac{1}{4}$ inch of the remaining trace not still connected to pin 18. You may wish to tin it with solder since it will later be soldered to.
4. (Underside) Connect pin 18 of ROM 'D8' to pin 12 of ROM 'EO' (ground)
5. (Underside) Connect pin 18 of ROM 'EO' to the trace which previously went to pin 18 of ROM 'D8' (and which should be pretinned if step 3 was followed).
6. (Underside) Connect pin 21 of ROM 'D8' to pin 21 of ROM 'DO'. Then connect both of these to pin 24 of either ROM (VCC).
7. Note that the $\overline{\text{INH}}$ control function (pin 32 on the APPLE-II I/O BUS connectors) will not disable the 2716 EROMs in the 'DO' and 'D8' ROM slots, since pin 21 is a power supply pin and not a chip select input on the EROMs.
8. 2716 EROM devices may now be used in sockets 'DO' and 'D8'.



ADDING COLORS TO APPLE'S HIGH-RESOLUTION

GRAPHICS HARDWARE

The Apple II normally provides a choice of black, white, green, or violet for hi-resolution graphics. However, the color encoding scheme is actually capable of specifying two additional colors: blue and orange. This application brief details a simple hardware modification which will allow the Apple II to display all six colors.

NOTE: THE INFORMATION HEREIN IS PROVIDED FOR THE CONVENIENCE OF THE SOPHISTICATED OWNER. ANY USER MODIFICATION TO THE APPLE II (INCLUDING THIS ONE) VOIDS THE FACTORY WARRANTY.

1. Remove the Apple II PC board from its enclosure

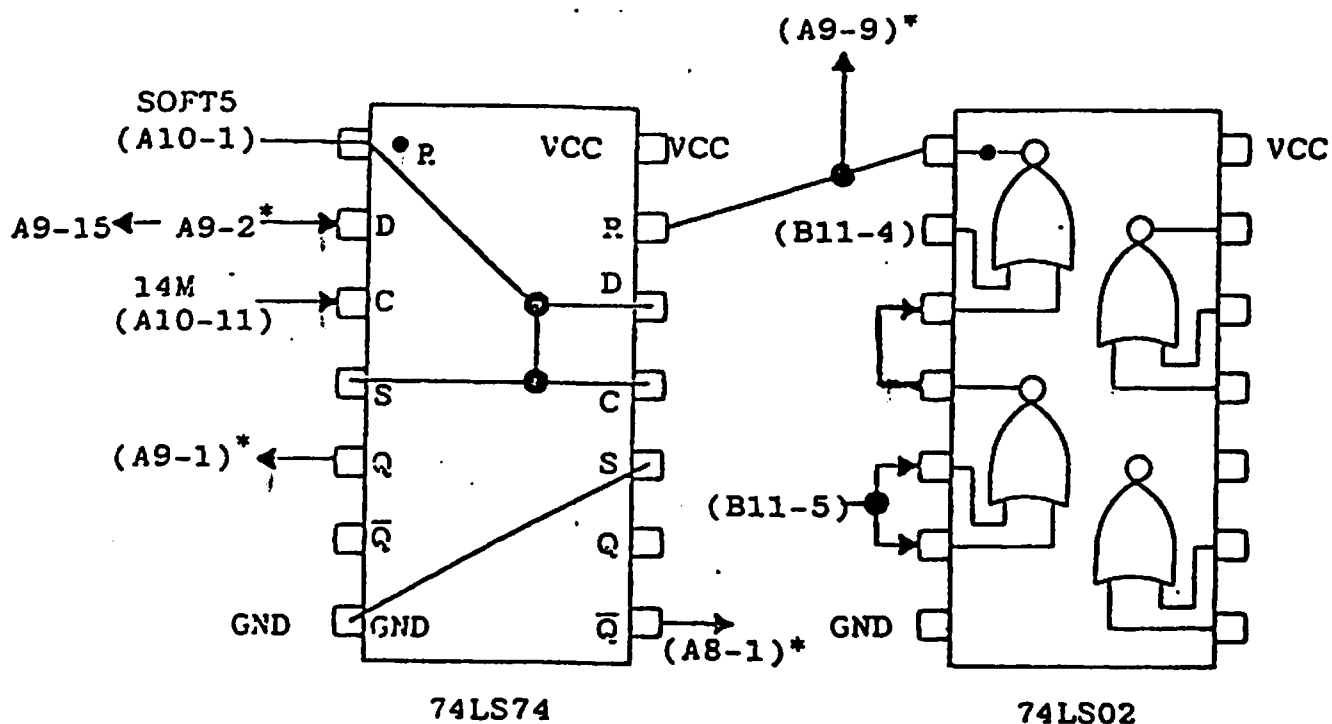
- (a) Remove the ten (10) screws securing the plastic top piece to the metal bottom plate. Six (6) of these are flat-head screws around the perimeter of the bottom plate and four (4) are round-head screws located at the front lip of the computer. All are removed with a phillips-head screwdriver. Do not remove the screws securing the power supply or nylon insulating standoffs...
- (b) Lift the plastic top piece from the bottom plate while taking care not to damage the ribbon cable connecting the keyboard to the PC board. This cable will have to be disconnected from one or the other.
- (c) Disconnect the power supply from the PC board.
- (d) Remove the #8 nut and lockwasher securing the center of the PC board. (These will not be found on the earlier Apple II computers.)
- (e) Carefully disengage each of 6 nylon insulating standoffs from the PC board. (7 on earlier versions)
- (f) Lift the PV board from the bottom plate.

2. Add in the new circuitry

- (a) Lift the following IC pins from their sockets.

A8-1	A9-1
A8-6	A9-2
A8-13	A9-9

- (b) Mount a 74LS74 (dual C-D flip-flop) and a 74LS02 (quad NOR gate) in the Apple II breadboard area (A11 to A14 region).
- (c) Wire the following circuit (* indicates that wiring is to a pin which is out of its socket).



(B8-14) → (A8-6)*
 (B8-7) → (A8-13)*

3. Reverse step 1 to reassemble the Apple II



APPLESOFT ARRAY ERASER

```
100 HOME
110 VTAB 5
120 PRINT TAB( 14);"ARRAY ERASER"
130 COSUB 10000
140 GOSUB 1050
150 PRINT "HERE ARE THE ARRAYS"
160 LIST 180
170 COSUB 1000
180 DIM A(100),B(100)
190 PRINT "AND HERE'S WHAT WE PUT IN THEM"
200 LIST 220 - 230
210 COSUB 1000
220 LET A(100) = 100
230 LET B(100) = 100
240 PRINT "OK, LET'S PRINT THEM OUT"
250 LIST 260 - 270
260 PRINT B(100)
270 PRINT A(100)
280 COSUB 1000
290 PRINT "NOW TO ERASE ARRAY 'A'"
300 LIST 310
310 CALL 768,A
320 COSUB 1000
330 PRINT "OK, NOW LET'S PRINT B(100) TO SHOW"
340 PRINT : PRINT "THAT IT'S STILL THERE"
350 LIST 360
360 PRINT B(100)
370 COSUB 1000
380 PRINT "NOW LET'S TRY TO PRINT A(100). THE"
390 PRINT : PRINT "ERROR WE GET PROVES THAT THE ARRAY IS"
400 PRINT : PRINT "GONE."
410 LIST 420
420 PRINT A(100)
430 END

1000 REM 'PRESS ANY...' ROUTINE
1010 VTAB 23
1020 PRINT TAB( 9);"PRESS ANY KEY FOR MORE"
1030 HTAB 20
1040 GET AS$
1050 VTAB 10
1060 HTAB 1
1070 CALL - 958: REM CLEAR SCREEN
1080 RETURN
10000 REM THE 'ERASE' POKER
10010 FOR J = 768 TO 823: READ K: POKE J,K: NEXT J
10020 RETURN
10030 DATA 32,177,0,32,217,247
10040 DATA 24,160,2,165,155
10050 DATA 133,66,113,155,133
10060 DATA 60,200,165,156,133
10070 DATA 67,113,155,133,61
10080 DATA 136,56,165,109,133
10090 DATA 62,241,155,133,109
10100 DATA 200,165,110,133,63
10110 DATA 241,155,133,110,160
10120 DATA 0,32,44,254,32,163
10130 DATA 217,76,152,217
```



```
1000 REM      HIRES SCRIN FUNCTION
1010 REM      DEMO
1020 REM
1030 REM LOAD IN BINARY STUFF
1040 REM
1050 FOR J = 768 TO 806: READ K: POKE J,K: NEXT J
1060 DATA 32,227,223,133,133
1070 DATA 132,134,169,208,32
1080 DATA 192,222,165,18,72
1090 DATA 165,17,72,32,185
1100 DATA 246,32,17,244,165
1110 DATA 48,49,38,240,2
1120 DATA 169,1,168,32,1
1130 DATA 227,76,91,218
1140 POKE 1013,76: POKE 1014,0: POKE 1015,3
1150 REM -----
1160 REM      DRAW SOMETHING
1170 REM -----
1180 LET HO = 120:VO = 60
1190 HGR
1200 HCOLOR= 3
1210 HPLOT HO,VO
1220 FOR J = 1 TO 10
1230 HPLOT TO RND (9) * 40 + HO, RND (9) * 40 + VO
1240 NEXT
1250 HOME
1260 VTAB 22
1270 FOR J = 0 TO 3000: NEXT
1280 REM -----
1290 REM      CONVERT IT TO LORES
1300 REM -----
1310 GR
1320 COLOR= 2: FOR V = 0 TO 39: HLIN 0,39 AT V: NEXT
1330 FOR V = 0 TO 39
1340 FOR H = 0 TO 39
1350 COLOR= 12: PLOT H,V
1360 REM -----
1370 REM      THIS IS IT !!!
1380 REM      THE SYNTAX IS:
1390 REM      &A=B,C  WHERE
1400 REM      A WILL GET THE 1 OR 0
1410 REM      B,C IS THE HIRES
1420 REM      COORDINATES AS IN
1430 REM      HPLOT
1440 REM -----
1450 & A = H + HO,V + VO
1460 COLOR= A * 15
1470 PLOT H,V
1480 NEXT
1490 NEXT
```



```
10 REM *****
20 REM *
30 REM * SIMPLE TONES FOR *
40 REM *
50 REM * APPLESOFT II *
60 REM *
70 REM * J CROSSLEY *
80 REM *
90 REM *****
100 REM
110 REM THIS IS THE ROUTINE
120 REM FROM PAGE 45 OF THE
130 REM 'RED BOOK' MODIFIED
140 REM FOR APPLESOFT.
150 REM
160 HOME
170 LIST - 150
180 REM
190 REM INSERT THE FOLLOWING
200 REM ROUTINE IN YOUR PROGRAM
210 REM THEN POKE 768, TONE
220 REM POKE 769, DURATION
230 REM AND, CALL 770 FOR TONES
240 REM
250 FOR I = 770 TO 790
260 :: READ J
270 :: POKE I, J
280 NEXT
290 DATA 173, 48, 192, 136, 208, 5
300 DATA 206, 1, 3, 240, 9, 202, 208
310 DATA 245, 174, 0, 3, 76, 2, 3, 96
320 FOR TN = 100 TO 10 STEP - 1
330 :: POKE 768, TN: REM TONE
340 :: POKE 769, 10: REM DURATION
350 :: CALL 770
360 NEXT
370 HOME
380 LIST 180, 360
```



```
100 REM          PRINT USING
110 REM
120 REM          SIMULATOR
130 REM
140 REM          AUG 79
150 REM
160 REM          J. CROSSLEY
170 REM
180 LET N = 2: REM SET NUMBER
190 REM          OF DECIMALS
200 LET S = 5: REM SET FIELD
210 REM          WIDTH
220 HOME
230 FOR X = - 5 TO 5 STEP .501
240 PRINT X,"$";
250 GOSUB 2000
260 PRINT
270 NEXT X
280 PRINT
290 PRINT "UNFORMATTED          FORMATTED"
300 END
1000 REM THIS IS THE FORMATTING
1010 REM SUBROUTINE. THE INPUT
1020 REM IS 'X','N', AND 'S'
1030 REM X IS THE NUMBER TO BE
1040 REM BE PRINTED
1050 REM N IS THE NUMBER OF
1060 REM DIGITS RIGHT OF '.'
1070 REM S IS THE WIDTH OF THE
1080 REM RIGHT JUSTIFIED
1090 REM PRINTING FIELD
1100 REM
2000 X$ = " " + STR$ ( INT (X * 10 ^ N + .5))
2010 Q = LEN (X$) - ( VAL (X$) < 0)
2020 PRINT SPC( S - Q * (Q > N + 1) - (N + 2) * (Q < = N + 1));
2030 PRINT MID$ (X$,1 + ( VAL (X$) < 0),(Q < = N) + (Q - N) * (Q > N))
;
2040 PRINT MID$ ("0.00",1 + ((N + 1) < Q),1 + (N - Q + 2) * (Q < N + 2)
);
2050 PRINT RIGHT$ (X$,N * (Q > N) + (Q - 1) * (Q < = N));
2060 RETURN
```



CONVERTING INTEGER BASIC PROGRAMS TO APPLESOFT

Page 76 of the DOS 3.2 Reference Manual contains a routine which can be used to convert Integer Basic programs to Applesoft using the disk system. Integer Basic is required for the first step of the conversion only.

First, load in the Integer program. Add the routine on page 76 at the very beginning of the program, adjust the LIST line for the specific program, and give the textfile a unique name. Then type "RUN". This will cause the program to be listed out to the disk as a textfile under the name specified. At this point, the text file is independent of the language.

To convert, type "EXEC <your filename>" in Applesoft. The textfile will be read into the system as if typed from the keyboard. Since Applesoft does not check syntax at input time, the Integer syntax is accepted, although the program may not run. Save the program in Applesoft as usual.

Micro Magazine, January 1980, contains an article describing Integer to Applesoft conversion using cassette instead of disk. The article also describes some of the syntax and logic changes that will usually have to be made.



APPLESOFT RANDOM NUMBERS

The Applesoft random number generator, like all such routines, is only a pseudo-random generator, so non-random patterns will eventually occur. The frequency of repetition of these patterns will vary from procedure to procedure. Proper re-seeding of the random number generator during a program will help prevent the appearance of large repeating sequences. This can be done in two ways, and for best results, both should be used.

1. Seed the random number calculation at the beginning of the program, using the keyboard count location. This will take the form

```
S = PEEK(78) + PEEK(79)*256  
X = RND(-S)
```

2. Within the random generating portion of the program, insert a statement of the form `Z = RND(-RND(9))`, which will begin a new random sequence. (See page 102 of the Applesoft Reference Manual.)

Please be aware that no method will completely eliminate patterns in the random numbers generated, but we can break up the sequences so that objectional non-random patterns are less likely to appear.



OUT OF MEMORY ERRORS

There are two ways to get an 'OUT OF MEMORY ERROR' in Applesoft. The most obvious cause is to have a program that is too big or uses too many variables. The only solution in this case is to trim down the program, keep the data on a disk, chain the program in from the disk in segments, or upgrade your system to 48K.

The less obvious cause is stack overflow. This is easy to spot because after getting the OUT OF MEMORY error, PRINT FRE(0) tells you that there is still free memory and, since the error clears the stack, everything else seems normal. The problem is that Applesoft uses the 6502 stack to save it's recursive subroutine calls and the stack is a limited resource. Here are some causes:

- * Too many levels of FOR-NEXT loop
- * Too many levels of GOSUB
- * Excessively complex mathematical or string formulas
- * GOSUBs with no RETURN
- * Improper recovery in ONERR GOTO routines
- * CALLs or interrupts that don't restore the stack properly

NOTE: THESE EFFECTS ARE CUMULATIVE. You might be affected by more than just one.

The first four are inherent in your program structure. If your program is cleanly structured then these probably won't cause trouble.

If you are using ONERR GOTO you should carefully examine pages 81, 82, and 136 of the Applesoft Reference manual. There are two correct ways to leave an ONERR routine. You can use RESUME which takes care of the stack and re-executes the statement that caused the error or you can use the stack recovery routine on pages 82 and 136 (the example on page 136 is easier to use) before you do a GOTO to a line number. Beware, this recovery routine does not clear any GOSUBs or FOR...NEXT loops!

When Applesoft executes a CALL, it does a 6502 JSR to the specified address. It's up to you to pull off anything you pushed on the stack before the RTS. Likewise the routine to handle an interrupt from a peripheral card must restore the stack and the registers.

A very large, complex program may be forced into a stack overflow condition by the user's responses. CALL 54915 will clear the stack without hurting the variables, but it wipes out all pending FOR-NEXT loops, GOSUBs and formulas. This allows a program controlled respart. Beware, this is no substitute for good programming practices but a way to recover when a program exceeds the Apple's capabilities.



VTAB AND HOME CONVERTER FOR THE M & R SUP-R-TERMINAL BY DAN SOKEL

Hello is the logon program and it spells out the limitations of the converter.

```
10 REM DOCUMENTATION
20 D$= CHR$(4)
30 PRINT D$;"PR#3"
40 PRINT D$;"MAXFILES 5":PRINT:PRINT D$
50 PRINT CHR$(140)
60 PRINT "This program converts VTAB statements in APPLESOFT PROGRAMS
70 PRINT "to the correct CTRL-SHIFT-N sequence for use with the M & R"
80 PRINT "Enterprises SUP-R-TERMINAL board."
90 PRINT
100 PRINT "The following limitations must be observed--"
110 PRINT "1. The program to be converted MUST be APPLESOFT."
120 PRINT "2. The program must be on this disk."
130 PRINT "3. The program cannot have any line numbers above 32749."
140 PRINT "4. There must be room on this disk for 2 ASCII copies of
    the program."
150 PRINT
160 PRINT "The conversion is not very fast---but it works."
170 PRINT "After the conversion is complete the new program is
    in memory."
180 PRINT "IT IS UP TO YOU TO SAVE IT -- Don.t forget."
190 PRINT
200 PRINT "Type return if you are ready to convert, space to exit."
210 GET A$
220 D$= CHR$(4): PRINT
230 IF A$= CHR$(13) THEN PRINT D$;"RUN FIXVTABS"
240 END
```

FIXVATABS is the starting point for converting - RUN FIXVTABS OR HELLO

```
10 D$= CHR$(4)
20 PRINT D$;"OPEN TEMP"
30 PRINT D$;"DELETE TEMP"
40 PRINT "ENTER NAME OF PROGRAM TO BE CONVERTED..";:INPUT A$
50 PRINT D$;"NOMON C,I,O"
60 PRINT D$;"EXEC XFERTOTEXT"
70 END
```


XFERTOTEXT is A TEXTFILE and contains the following data:

```
PRINT DS;"LOAD ";AS
32750 DS=CHR$(4)
32751 PRINT DS;"OPEN TEMP"
32752 PRINT DS;"DELETE TEMP"
32753 PRINT DS;"OPEN TEMP"
32754 PRINT DS;"WRITE TEMP"
32755 PRINT LIST 0-32749
32756 PRINT CHR$(27) +CHR$(27) +CHR$ +CHR$(27)
32757 PRINT DS;"CLOSE TEMP"
RUN 32750
RUN CONVERT
```

CONVERT does the work - here.s how:

```
10 DS=CHR$(4)
20 PRINT DS;"NOMON C,I,0"
30 PRINT DS;"OPEN NEWPROG"
40 PRINT DS;"DELETE NEWPROG"
50 PRINT DS;"OPEN NEWPROG"
60 PRINT DS;"OPEN TEMP"
70 PRINT DS;"READ TEMP"
80 GET AS
90 CS=CS + AS
100 IF MID$(CS,2,3) = CHR$(27)+CHR$(27) + CHR$(27) THEN GOTO 340
110 IF AS < > CHR$(13) THEN GOTO 80
120 FOR I = 1 TO LEN (CS)
130 IF MID$(CS,I,4)="VTAB" THEN GOSUB 200
140 IF MID$(CS,I,4)="HOME" THEN GOSUB 400
150 IF MID$(CS,I,10)="CALL-936" THEN GOSUB 380
160 BS=BS + MID$(CS,I,1)
170 NEXT I
180 PRINT DS:PRINT DS;"WRITE NEWPROG": PRINT BS
190 BS=" ":CS = " ": GOTO 70
200 REM CONVERT VTABS HERE
210 P=0
220 XS="PRINT CHR$(158)+CHR$ (32+POS(0)) + CHR$(32+"
230 I = I +4
240 BS=BS + XS
250 FOR K = I TO LEN (CS)
260 IF MID$(CS,K,1) = CHR$(13) THEN GOTO 300
270 IF MID$(CS,K,1) < > ":" THEN BS=BS+MID$(CS,K,1): P=P+1
280 IF MID$(CS,K,1)=":" THEN GOTO 300
290 NEXT K
300 I= I + P
310 BS=BS + ");"
320 PRINT DS: PRINT "***** FOUND VTAB *****";
330 RETURN
```

```
340 PRINT D$;"CLOSE"  
350 TEXT  
360 PRINT D$;"EXEC TEXTOPROG"  
370 END  
380 REM FOUND CALL -936  
390 I= I + 6  
400 REM FOUND HOME  
410 I = I + 4  
420 X$="PRINT CHR$(140);"  
430 B$= B$ + X$  
440 PRINT D$;: PRINT "***** FOUND HOME *****";  
450 RETURN
```

Lastly, the texfile TEXTOPROG recreates the program in memory

```
NEW  
D$ = CHR$(4)  
PRINT D$; "EXEC NEWPROG"
```



MODIFYING THE LISA ASSEMBLER

TO HANDLE USER ROUTINES

Randall Hyde

April 7, 1980

If you enjoy your privacy, one of the worst things you can do is write a piece of systems software and market it. For every little bug (or "feature" as I prefer to call them), you're pretty much guaranteed ten phone calls and 100 letters. Having authored the LISA interactive assembler, I can say from experience that the support required of a high level product is tremendous. I doubt that the royalties received for my product have come anywhere near paying me back for the time spent creating and improving it. Now don't get me wrong, I enjoy what I am doing. Writing (and improving) LISA was strictly a hobby undertaking, and I certainly don't mind the chance to talk to fellow Apple owners across the nation. But nevertheless, as the old saying goes, you can't please all of the people all of the time. No matter what neat new feature a person adds to a product, there will always be someone who desires something additional. This has been especially true for the LISA assembler since the first day it was released. Most requests for changes to LISA have been quite reasonable and, in fact, have been implemented. Some requests, however, are of such limited use that they did not warrant inclusion into LISA. LISA is big enough as it is. There is really no need to make it gigantic just to satisfy everyone. The purpose of this article is to describe how to modify LISA so that special purpose functions may be included. I will assume that you are using LISA V1.5D, but this discussion applies to version 1.5C and to a limited extent to LISA version 2.0 as well.

Before describing how to "customize" LISA, the reader must first have an understanding of the internal structure of LISA: both in terms of the internal workings and an understanding of how source code is stored in memory. An understanding of the source code format for LISA is required in order to write routines which search out labels and perform other related functions. An understanding of the internal workings of LISA is required in order to patch into it to make the desired changes.

Internally, LISA consists of five modules. A command interpreter reads a line of text and performs the desired task. A collection of routines perform the LIST, DELETE, MODIFY, WRITE, etc., commands. A third section of LISA inputs a line of code, checks for proper syntax, and then tokenizes the code. The fourth section is the actual assembler which converts the source code to object code. The fifth section is an I/O executive which handles all I/O in LISA.

The most important module to discuss is the I/O module. Since without I/O, nothing else can be performed. LISA's I/O drivers are an enhancement over those provided by the Apple monitor. For our purposes, two things must be considered when using LISA's I/O. First, all characters are converted to normal ASCII format. That is, the high order bit of each character is turned off. Second, the line input routine stores its data starting at location \$300, instead of \$200, as is the case with the Apple monitor routine. This is due to the fact that Apple DOS uses page two for collecting DOS commands and as such it would often wipe out an input line. Lines of text in LISA can be used when writing your own commands for the LISA command interpreter. The internal format for a LISA source statement is described in sufficient detail in the LISA documentation manual, so I will not repeat its description here.

So, how does one interface to the LISA command interpreter? Well, when designing LISA I realized that there was no way that I could foresee what types of peripherals would be used with the LISA assembler. So I incorporated a user defineable function for purposes of invoking user defined printer interfaces etc. Now it just so happens that this user defineable function can be used to implement other user defined functions as well as printer drivers. The remainder of this article is dedicated to exploring this particular aspect of LISA.

In most of the letters I've received asking for additional features, the two features requested the most were the ability to execute Apple monitor commands directly from the LISA command interpreter ("just like the SC-Assembler II" more than one user has mentioned--was that a threat?!!) and the ability to run machine language programs directly from LISA. Despite the large number of users requesting these additions I have not incorporated these features for two very good (I feel) reasons. In the former case, I feel it's just as easy to hit the reset key to get into the Apple monitor as it is to type some special symbol such as "\$" (as with the Apple mini-assembler) in order to perform an Apple monitor command. The latter request - allowing the user to execute a machine language program directly from LISA and then return to LISA is somewhat dangerous. Many user programs use some of the zero page locations where important pointers are kept. By allowing the user to run a program directly from LISA it allows the programmer to get lazy and not save the program onto disk before running it. Forcing the user to press reset, or use the LISA "BRK" command makes them think twice before destroying possible pointers.

Nevertheless, there are several folks out there who tell me all the time "Give me the rope; if I hang myself it will be my own fault." Who can argue with that? As such the following LISA auxiliary program was born. This short little program, which resides from location \$6F00 to \$6FFF, implements the previously described functions (Apple monitor commands and the ability to run a program from LISA) as well as leaving room open for user defineable routines. Its usage revolves around the control-P user function in LISA. Briefly, a jump to sub-routine is made to location \$7009 whenever the first non-blank character on a line is a control-P. Normally at location \$7009 is a jump back to the LISA command processor. By writing over the jump address

with a user defined address, it is possible to direct program flow to a routine to perform additional functions. Therein lies the method of expanding LISA in a manner decided upon by the user.

The only problem remaining is where does this user defined routine go? Under normal circumstances, LISA uses up (for all practical purposes) all of the available space in a 48K machine. Once again, LISA's flexibility allows the user to reserve some memory for his own use. In a normal LISA system, locations \$1800 through \$5800 are reserved for textfile storage, locations \$5800 through \$6FFF are reserved for Symbol table useage and locations \$7000 through \$BFFF are reserved for LISA and DOS. Locations \$7022 & \$7023 in LISA contain the last address available for symbol table useage. Normally, these two bytes contain the value \$6FF7. By changing this to \$6EFO, it is possible to reserve the memory space from location \$6FOO through \$6FFF for use by the user routine. Setting up this address, as well as modifying the USRJMP address in LISA is handled in lines 30 - 41 of the following routine. By simply BRUNing the routine which follows, not only will it be loaded into memory properly, but it will also protect itself from being walked on by LISA, as well as setting up the correct jump vectors.

At line 77 is the beginning of the routine which is called when the control-P command is detected. Now whenever the program gets to this point, the 6502 Y register contains an index into page three which points at the control-P character. By incrementing the Y register by one, the Y register will point to the next available character in the LISA input buffer. At line 65 the subroutine BLKDEL is called. This routine increments the Y register as long as it is pointing at a blank character. This allows interleaving blanks to be typed in without causing any problems. Upon return from BLKDEL, the 6502 accumulator contains the first non-blank character detected. At this point, we can compare this character with several "test" characters to determine which routine we wish to execute. In this program, any line beginning with a "\$" means that the following text is to be interpreted as an Apple monitor command. If the line begins with an "R" then the program runs the code at the address (symbolic or non-symbolic) specified in the following address expression.

Beginning at line 95 is the code to handle the Apple monitor command request. First the Y index register is incremented by one so that it points to the first character past the "\$". Next, this code is OR'd with \$80 to convert it to a form acceptable by the Apple monitor and stored into page two. Finally the monitor command is executed by executing the "FAKEMON" code found in the Apple mini-assembler.

At line 145, the RUN command is checked for. If the first non-blank character after the control-P is an "R", then FNDBLK is used to find the next blank on the line. Once this is accomplished, the following blanks are skipped over with a call to BLKDEL. Finally, after some initial setup, GETADR is called and looks up the address on the remainder of the line. Any valid LISA address is allowed, as are symbolic references. If the address is not found, then an error is reported, and the user is returned to LISA. Otherwise the code at the address specified by the user is executed.

If the first non-blank character on a line is neither a "\$" nor an "R" then this routine drops down to USRCMD which simply returns to LISA. If you wish to incorporate your own commands into LISA you may add them at this point. Just keep in mind that there are only 64 bytes left before \$7000 gets clobbered, so you may have to re-ORG this routine.

To wrap things up, it would be wise to give a few examples of how to use this auxillary routine since this article did not provide a very good description.

TO EXECUTE AN APPLE MONITOR COMMAND FROM LISA:

SYNTAX: (control-P) \$ (and Apple monitor command)

EXAMPLES:

(control-P) \$800L

(control-P) \$800G

(control-P) \$7F8:00

TO RUN A PROGRAM FROM THE LISA COMMAND LEVEL:

SYNTAX: (control-P) R (address expression)

Note: At least one blank must appear between the "R" and the address expression.

(control-P) R START

(control-P) R START+3

(control-P) R \$800

Obviously, any user command has its own syntax.

Hopefully these examples have demonstrated how one may easily modify LISA and incorporate any special command you so desire. With a little work, it is possible to write label search routines, special driver routines, etc. The implimentation of these functions will be left to the reader.

6F00	1	ORG \$6F00	
6F00	2	; APPLIC MONITOR EQUATES	
6F00	3		
6F00	4	GETNUM EQU \$FFA7	
6F00	5	TOSUB EQU \$FFBE	
6F00	6	ZMODE EQU \$FFC7	
6F00	7	CHRTBL EQU \$FFCC	
6F00	8	BL1 EQU \$FE00	
6F00	9	PAGE2 EQU \$200	
6F00	10		
6F00	11	YSAV EPZ \$34	
6F00	12	MODE EPZ \$31	
6F00	13		
6F00	14	; LISA EQUATES	
6F00	15		
6F00	16	USRJMP EQU \$7009	;USER FUNCTION JMP ADDRESS
6F00	17	SYND EQU \$7022	;ADDRESS OF END-OF-SYMBOL TABLE
6F00	18	GETADR EQU \$83F7	;LISA ADDRESS EPRESSION ANALIZER
6F00	19	PRINT EQU \$7983	;LISA PRINT ROUTINE
6F00	20	SYMAIR EPZ \$81	;ADDRS IS RETURNED HERE.
6F00	21	ADRID EPZ \$8F	;\$40 RETURNED IF ADDRESS FOUND
6F00	22	PNTR EPZ \$6A	;POINTER TO ADDRESS EXPRESSION
6F00	23		
6F00	24	INPUT EQU \$300	;LISA INPUT LINE BUFFER
6F00	25		
6F00	26		
6F00	27		
6F00	28		
6F00	29	; INSTALLATION SECTION.	
6F00	30		
6F00	31	; BY BRUNNING THIS PROGRAM, IT WILL AUTO-	
6F00	32	; INSTALL ITSELF AND SET UP THE LISA	
6F00	33	; PRINTERS SO THAT IT WILL NOT GET	
6F00	34	; WALKED UPON	
6F00	35		
6F00	36	INSTAL:	
6F00	37		
6F00	A96E	38 LDA #\$61	;INIT H.O. BYTE OF MAXIMUM SYMBOL.
6F02	8D370	39 STA SYMEND+1	;TABLE ADDRESS
6F05	A9F0	40 LDA #\$F0	;INIT L.O. BYTE OF MAXIMUM SYMBOL.
6F07	8D270	41 STA SYMEND	;TABLE ADDR
6F0A		42	
6F0A		43	; THE FOLLOWING CODE SETS UP THE END-OF-SYMBOL
6F0A		44	; TABLE POINTER SO THAT THE SYMBOL.
6F0A		45	; TABLE DOES NOT WIPE OUT THESE ROUTINES
6F0A		46	
6F0A	A915	47 LDA #START	
6F0C	8D0A70	48 STA USRJMP+1	
6F0F	A96F	49 LDA #START	
6F11	8D0B70	50 STA USRJMP+2	
6F14	60	51 RTS	
6F15		52	
6F15		53	
6F15		54	

```

6F15      55 ;
6F15      56 ;;
6F15      57 ; THIS ROUTINE, WHEN INVOKED BY A CONTROL-P,
6F15      58 ; WILL DO SEVERAL THINGS.
6F15      59 ;
6F15      60 ; 1) If the CONTROL-P is followed by a "$" then the
6F15      61 ;     remaining data on the line will be treated as an
6F15      62 ; APPLE II Monitor command.
6F15      63 ;     If the CONTROL-P is followed by an "R"
6F15      64 ; then the rest of the line is treated as
6F15      65 ; an address expression and the routine
6F15      66 ; at that address is executed.
6F15      67 ; Note: If symbolic addresses are used
6F15      68 ;     then this command must be
6F15      69 ;     used after an assembly.
6F15      70 ;
6F15      71 ;
6F15      72 ; If none of these conditions are met, then a
6F15      73 ; branch to a user definable routine
6F15      74 ; is made.
6F15      75 ;
6F15      76 ;
6F15      77 START:
6F15      78 ;
6F15 D8    79      CLD
6F16      80 ;
6F16      81 ; AT THIS POINT, THE Y REGISTER IS
6F16      82 ; AN INDEX INTO THE INPUT BUFFER AND
6F16      83 ; CURRENTLY POINTS AT THE CONTROL-P.
6F16      84 ;
6F16 C8    85      INY                ;MOVE TO THE NEXT CHARACTER
6F17 20A6CF 86      JSR BLKDEL        ;DELETE ALL LEADING BLANKS
6F18 C9 87      CMP C '$'          ;IS THE NEXT NON-BLANK A "$"?
6F19 0042 88      BNE TSTRUN        ;GO TO TEST FOR RUN ROUTINE
6F1E      89 ;
6F1E      90 ; IF SO, TRANSFER DATA THROUGH THE RETURN
6F1E      91 ; TO THE APPLE INPUT BUFFER AND CONVERT
6F1E      92 ; TO APPLE ASCII CODE.
6F1E      93 ;
6F1E      94 ;
6F1E C8    95      INY                ;MOVE PAST THE "$"
6F1F A200 96      LDX #0            ;SET UP INDEX INTO PAGE TWO.
6F21      97 LOOP:
6F21 B90003 98      LDA INPUT,Y      ;GET DATA FROM LISA INPUT BUFFER
6F24 0930 99      ORA #$80          ;CONVERT TO APPLE'S BRAND OF ASCII
6F26 9D0002 100     STA PAGE2,X      ;SAVE IN APPLE INPUT BUFFER
6F29 C98D 101     CMP #$8D          ;QUIT WHEN RETURN IS FOUND
6F2B 1004 102     BEQ LPXIT
6F2D E8 103     INX
6F2F 08 104     INY
6F2F D0F0 105     BNE LOOP
6F31      106 ;
6F31 A000 107     LPXIT LDY #0        ;SET UP INDEX FOR APPLE MONITOR.
6F33 4C3B61 108     JMP IAKMON
6F36      109 ;
6F36      110 ;
6F36      111 ; SIMULATE APPLE MONITOR
6F36      112 ; THIS CODE WAS RIPPED OFF OUT OF THE APPLE
6F36      113 ; MINI ASSEMBLER.
6F36      114 ;
6F36      115 ;

```


6F36	20BEFF	116	FKMN3	JSR TOSUB	
6F39	A424	117		LDY YSAV	
6F3B		118			
6F3B	20A7FF	119	FAK1 ON	JSR GETNUM	
6F31	8434	120		STY YSAV	
6F40	A017	121		LDY #17	
6F42		122			
6F42	88	123	FLMN2	DEY	
6F43	3012	124		BMI RESETZ	
6F45	D900FF	125		CMP CHRTBL,Y	
6F48	110F8	126		BNE FKMN2	
6F4A	C015	127		CPY #15	
6F4C	110E8	128		BNE FKMN3	
6F4E	A531	129		LDA MODE	
6F50	A000	130		LDY #0	
6F52	C634	131		DEC YSAV	
6F54	200011	132		JSR BL1	
6F57		133			
6F57	A98D	134	RESETZ	LDA #8D	
6F59	20F11D	135		JSR \$FDED	
6F5C	20ED11	136		JSR \$FD10	
6F5F	60	137		RTS	
6F60		138			
6F60		139			
6F60		140			
6F60		141			
6F60		142			
6F60		143			
6F60		144			
6F60		145	TST1 ON:		
6F60		146			
6F60	C952	147		CMP #'R'	
6F62	11041	148		BNE USRCMD	
6F64		149			
6F64		150			
6F64		151			
6F64	20B161	152		JSR FNDBLK	
6F67		153			
6F67		154			
6F67		155			
6F67	20A66F	156		JSR BLKDEL	
6F6A		157			
6F6A		158			
6F6A	A900	159		LDA #INPUT	;SET UP PNTR WITH
6F6C	856A	160		STA PNTR	;THE ADDRESS OF INPUT
6F6E	A903	161		LDA /INPUT	;DATA.
6F70	856B	162		STA PNTR+1	
6F72		163			
6F72		164			
6F72	201143	165		JSR GETADR	;ANALIZE ADDRESS EXPRESSION
6F75	248F	166		BIT ADREND	;SEE IF ADDRESS WAS FOUND
6F77	7023	167		BVS RUN1	;IF NOT, QUIT GRACEFULLY
6F79	208378	168		JSR PRINT	
6F7C	0D0D	169		HEX 0D0D	
6F7E	09CCCC	170		ASC "ILLEGAL ADDRESS EXPRESSION"	
6F81	C5C7C1				
6F84	CCA0C1				
6F87	C4C4D2				
6F8A	C5D3113				
6F8D	A0C508				
6F90	110D2C3				

```

6F93 D303C9
6F94 CFCE
6F95 0D0D00 171      HEX 0D0D00
6F96 60      172      RTS
6F9C      173      ;
6F9C      174      ;
6F9C      175      ; ADDRESS EXPRESSION WAS VALID, SO
6F9C      176      ; GO TO THAT ADDRESS AND EXECUTE
6F9C      177      ;
6F9C 20A26F 178      RUN1   JSR JSRIND
6F9F 4C0370 179      JMP $7003
6FA2      180      ;
6FA2 6C8100 181      JSRIND JMP (SYMAOR)
6FA5      182      ;
6FA5      183      ;
6FA5      184      ;
6FA5      185      ; ADDITIONAL USER STUFF CAN GO HERE
6FA5      186      ;
6FA5 60      187      USR CMD RTS
6FA6      188      ;
6FA6      189      ;
6FA6      190      ;
6FA6      191      ; BLKDEL- DELETES ALL LEADING BLANKS
6FA6      192      ;
6FA6 B90003 193      BLKDEL LDA INPUT,Y
6FA7 C920    194      CMP #' '
6FAB DC03    195      BNE BLKXIT
6FAD C8      196      INY
6FAE D0F6    197      BNE BLKDEL
6FB0      198      ;
6FB0 60      199      BLKXIT RTS
6FB1      200      ;
6FB1      201      ;
6FB1      202      ; FNDBLK- FINDS THE NEXT BLANK CHARACTER
6FB1      203      ;
6FB1      204      FNDBLK:
6FB1      205      ;
6FB1 B90003 206      LDA INPUT,Y
6FB4 C920    207      CMP #' '
6FB6 F007    208      BEQ FNDXIT
6FB8 C90D    209      CMP #$D
6FBA F003    210      BEQ FNDXIT
6FBC C8      211      INY
6FBD D0F2    212      BNE FNDXIT
6FBD      213      ;
6FBD 60      214      FNDXIT RTS
6FC0      215      ;
216      END

```

***** END OF ASSEMBLY

```

*****
*
* SYMBOL TABLE -- V 1.5 *
*
*****

```

LABEL, LOC. LABEL, LOC. LABEL, LOC.

**** ZERO PAGE VARIABLES:**

YSAV 0034 IBIDE 0031 SYHADR 0081 ADREND 008F PNTR 006A

**** ABSOLUTE VARIABLES/LABELS**

GETNUM 1FA7									
TOSLI FFBC	ZMODE FFC7	CHRTBL FFCC	BL1 FE00	PAGE2 0200	USRJMP 7009				
SYND 022	GETADR 3 39	PRINT 7 33	INPUT 0E00	INSTAL 6F00	START 6F15				
LOOP 6F21	LPXIT 6F31	FKMN3 6F36	FAKMON 6F3B	FKMN2 6F42	RESETZ 6F57				
TS1RUN 6F60	RUN1 6F7C	JSRIND 6FA2	USPCMD 6FA5	BLKDEL 6FA6	BLKXIT 6FB0				
FNDBLK 6FB1	FNDXIT 6FBF								

SYMBOL TABLE STARTING ADDRESS:6000

SYMBOL TABLE LENGTH:0112



PROGRAMMA International, Inc.

THIS ADDENDUM SUPPLEMENTS INTEGER BASIC/MACHINE LANGUAGE SOFTWARE DOCUMENTATION

How about saving the program on the supplied cassette onto disk? It's actually very simple to do if you follow these instructions carefully:

- 1) Type in the machine language program (on page 2) beginning at hexadecimal address \$1100.

NOTES:

- A) Simply type the numbers. The disassembled listing with comments is to assist you.
 - B) Any value can be substituted for ?? for now as it will be changed in step 3.
 - C) The routine can be found at the beginning of appropriate programs in the DEALER DEMO PACK.
- 2) Load in the program from cassette beginning at address \$1200. For example, if the normal load is "200.2000R", then load it "1200.3000R".
 - 3) Now take the high byte of the ending address of the normal load (i.e. \$20 for \$2000, \$40 for \$4000) and add \$0F to it (\$2F, \$4F...). Place this value into memory location \$111F.
 - 4) BSAVE the appropriate region of memory:
(A\$1100, L\$xx00 where xx=1F, 3F...)
 - 5) If there are any more programs you wish to store onto disk, repeat steps 2 thru 4.
 - 6) To execute a program, simply type "BRUN filename" and press return.

SUMMARY TABLE:

! NORM LOAD !	NEW LOAD	! HI !	!+OF !	L\$! SYS !
! 200.2000R !	1200.3000R	! 20 !	2F	! 1F00 !	24K !
! 200.4000R !	1200.5000R	! 40 !	4F	! 3F00 !	32K !
! 200.6000R !	1200.7000R	! 60 !	6F	! 5F00 !	48K !
! 200.8000R !	1200.9000R	! 80 !	8F	! 7F00 !	48K !

Here is the Machine Language program:

1100-	20 84 FE	JSR	\$FE84	SET NORMAL VIDEO
1103-	20 2F FB	JSR	\$FB2F	SET TEXT FULL WINDOW
1106-	20 93 FE	JSR	\$FE93	SIMULATE PR#0 (COUT)
1109-	20 89 FE	JSR	\$FE89	SIMULATE IN#0 (KEYIN)
110C-	A9 00	LDA	#\$00	
110E-	85 3C	STA	\$3C	FROM LO ADDR
1110-	85 42	STA	\$42	TARGET LO ADDR
1112-	A9 12	LDA	#\$12	
1114-	85 3D	STA	\$3D	FROM HI ADDR
1116-	A9 02	LDA	#\$02	
1118-	85 43	STA	\$43	TARGET HI ADDR
111A-	A9 FF	LDA	#\$FF	
111C-	85 3E	STA	\$3E	TO LO ADDR
111E-	A9 ??	LDA	#\$??	MAKE XX+\$0F OPERAND
1120-	85 3F	STA	\$3F	TO HI ADDR
1122-	A0 00	LDY	#\$00	
1124-	A9 02	LDA	#\$02	BY PUSHING \$02 & \$1F
1126-	48	PHA		ONTO THE STACK, WE SET
1127-	A9 1F	LDA	#\$1F	UP A PSEUDO RTS ADDR
1129-	48	PHA		
112A-	4C 2C FE	JMP	\$FE2C	DO MOVE, THEN DO "220G"

This routine will work on an Apple II Plus, although the program the routine operates upon may not.



THE APPLE II CASSETTE INTERFACE

INTRODUCTION:

This note is to explain the cassette interface built into the Apple II and Apple II plus. Included is information on the format and use of the read and write subroutines. It is assumed in this note that the cassette recorder is in the proper mode, play or record, when the read and write routines are executed. Also note that the timing is approximate and may vary from one Apple to another.

RECORDS:

A record is a block of binary data. This data may be a BASIC or APPLESOFT program, a machine language program, or just binary data. Records representing BASIC or APPLESOFT programs are really two records, the length of the program and the actual program. A record consists of a header, sync bit, the actual data, and a checksum byte for error detection.

MONITOR RECORD FORMAT

```

+-----+-----+-----+-----+
! HEADER !S!          DATA          !C!
+-----+-----+-----+-----+

```

BASIC PROGRAM RECORD FORMAT

```

+-----+-----+-----+-----+-----+-----+-----+
! HEADER !S! LB !C! HEADER !S!          PROGRAM          !C!
+-----+-----+-----+-----+-----+-----+-----+

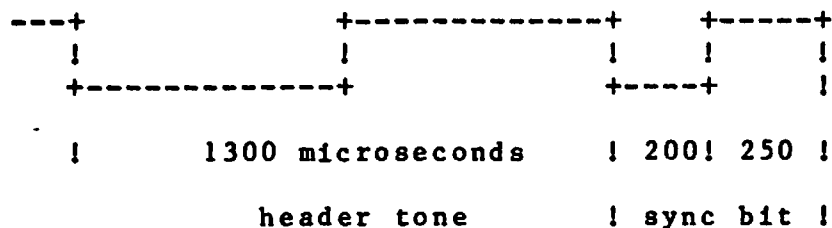
```

KEY: S = SYNC BIT
C = CHECKSUM BYTE
LB = BASIC PROGRAM LENGTH

THE HEADER:

The header consists of 10 seconds of 770 Hz tone, (1 cycle equals 1300 microseconds). This is enough time for the cassette motor to get up to speed and the plastic tape leader to go by. There is also a shortened header between the BASIC length bytes and the BASIC program itself. This header is generated by a subroutine called HEADR. The value of the accumulator on entry controls the length of the header tone. This can vary from 0.2 seconds to 40 seconds. On entry the X register should be 0 and the carry flag should be set. HEADR also generates a sync bit at the end of the tone. HEADR resides at hexadecimal address \$FCC9, or decimal address -882.

THE LAST CYCLE OF HEADER TONE AND SYNC BIT

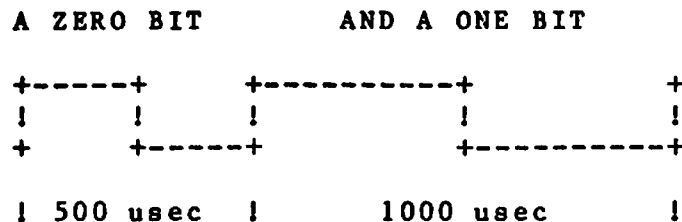


THE SYNC BIT:

The sync bit is one half cycle of 2500 Hz, (200 microseconds) and one half cycle of 2000 Hz, (250 microseconds). It is used to signal the end of the header tone and the start of the data. It is generated by HEADR.

THE DATA:

The data is recorded on the tape with a low starting address and a high ending address. Each byte of data is shifted out most significant bit first, least significant bit last. A zero bit is made up of one cycle of 2 kHz, (250 microseconds per half cycle) and a one bit is one cycle of 1 kHz, (500 microseconds per half cycle). This works out to 2000 baud for zeros only and 1000 baud for ones, an average of 1500 baud.



THE CHECKSUM:

All during reading or writing each data byte is EXCLUSIVE OR-ed with a checksum byte. This byte is written on the tape at the end of the data block. If the checksum computed during a read agrees with the checksum that was written out then the data is probably good. This method will detect an odd number of errors for any of the eight bits of the byte.

WRITING DATA:

The cassette output circuitry is quite simple. It is a flip-flop connected through a voltage divider to the jack on the back panel of the Apple. Any time the address \$C020 is accessed this flip-flop changes state. Accessing the flip-flop once every 500 microseconds generates a 1000 Hz tone.

READING DATA:

The cassette input circuit more complicated. It consists of a 741 operational amplifier configured as a zero crossing detector. That means that whenever the voltage at the input jack goes from positive to negative (or negative to positive) the output of the amplifier switches from a 1 to a 0 (or 0 to 1). The detector is accessed by any read to address \$C060. The sign bit (most significant bit) reflects the detector status. The read routines continually EXCLUSIVE ORs this bit with the value most recently read to detect a change in state. The amount of time required to change state indicates the incoming frequency which then is used to determine if a one or a zero has been recieved. After detecting the first zero crossing at the start of a read, the read routine uses HEADR to generate a 3.5 delay then waits for the sync bit. It then reads the data and puts it in the specified memory range.

USING THE CASSETTE INTERFACE:

To either read or write all you need do is specify an address range and execute the read or write subroutine. The address range is stored in four bytes, two for the start and two for the end. In both cases the least significant byte is first.

FROM THE MONITOR:

If the start is \$800 and the end is \$9FF then

800.9FFW will write the data to the cassette and
800.9FFR will retrieve it.

FROM MACHINE LANGUAGE:

Again if the start is \$800 and the end is \$9FF then store the address range,

```
LDA #$00
STA $3C      starting address low
LDA #$08
STA $3D      starting address high
LDA #$FF
STA $3E      ending address low
LDA #$09
STA $3F      ending address high
```

then JSR \$FEDC to write the data to the cassette or JSR \$FEFD to read from the cassette.

FROM BASIC:

First set up the address range. If S = the start and E = the end then from integer BASIC,

```
POKE 60,S MOD 256
POKE 61,S / 256
POKE 62,E MOD 256
POKE 63,E / 256
```

or from APPLESOFT,

```
POKE 60,S - INT(S / 256) * 256
POKE 61,S / 256
POKE 62,E - INT(E / 256) * 256
POKE 63,E / 256
```

Then to write out to cassette CALL -307 or to read in from the cassette CALL -259.



DEL CHARACTER KILLER 5 FEB 80

INTRODUCTION:

This routine won't allow the DEL character (underline) through the input routine. DOS 3.2 is required to use it.

SOFTWARE ENTRY

First you must decide which slot the interface will go in and enter the routine. The routine is customized for this slot number and won't work if used with a different configuration. Enter the routine using the values from the table for words in brackets, < >.

SLOT	1	2	3	4	5	6	7
CODE	C1	C2	C3	C4	C5	C6	C7

	COMMUNICATIONS	SERIAL
TYPE	07	05

Enter the monitor with CALL -155 and type

```
390:A9 <SLOT>
:20 8B FE
:A9 A0
:85 38
:A9 03
:85 39
:4C EA 03
:20 <TYPE> <CODE>
:C9 FF
:F0 F9
:60
```

To check your typing, type

390L

and compare your listing to the one below for slot 2 and a Communications Interface.

```
0390-  A9 02      LDA  #$02
0392-  20 8B FE   JSR  $FE8B
0395-  A9 A0      LDA  #$A0
0397-  85 38      STA  $38
0399-  A9 03      LDA  #$03
039B-  85 39      STA  $39
039D-  4C EA 03   JMP  $03EA
03A0-  20 07 C2   JSR  $C207
03A3-  C9 FF      CMP  #$FF
03A5-  F0 F9      BEQ  $03A0
03A7-  60         RTS
```

Now return to BASIC with 3DOG

SAVING THE PROGRAM TO DISK:

The routine must be in memory before the printer can be used with the delay. Save the routine by typing
BSAVE DEL FILTER, A\$390, L\$18

USING THE INTERFACE:

To use the interface you must first load the routine and initialize the interface. From command mode type

BLOAD DEL FILTER

CALL 912

This may be done from a program by entering
100 PRINT D\$;"BLOAD DEL FILTER" : CALL 912
assuming that D\$ is a control-D.

If you want to switch back to the video monitor for output type
IN#0

or in a program enter

200 PRINT D\$;"IN#0"

Then to reconnect the routine, all that is required is

CALL 917 or from a program

300 CALL 917



CORRECTIONS TO SUP-R-TERMINAL PRELIMINARY MANUAL By John Wilbur

VTABS

The preliminary manual has a number of errors. One of the more significant ones is the replacement for VTABS on page 14 under ABSOLUTE CURSOR POSITIONING. Control T Shift N should be replaced with Control Shift N (the Control T was wrong). In other words, the VTABS in your Basic programs should be replaced with:

```
PRINT CHR$(158);PRINT$(HOR);PRINT$(VERT)
```

Where HOR is the horizontal column desired + 32, and where VERT is the vertical line desired + 32.

To do the goto XY from the keyboard, you would type Control Shift N Key 1 - Key 2. Where Key 1 is the key you press to give you the horizontal position of the cursor (column 0-79). For example, if you wanted the cursor to move to column 18, you would find "18" in the rightmost column of appendix A labeled "SUP-R-TERMINAL MEANING". The key you would press would be the "2" key (shown in the column labeled "type") just to the left of the 18.

PASCAL NOTE

SUP-R-TERMINAL works best in PASCAL. We have received some feedback from PASCAL users that say the KEYPRESS function in their programs requires a program modification in order to work properly. Keypress is a function that checks the keyboard to see if you have pressed a key before allowing the program to continue. The PASCAL system looks at slot three to see if there is an external terminal connected. If it sees our card plugged in, it assumes there is, and then doesn't check the Apple keyboard for the keypress function. The best solution is to add a PEEK & POKE function to the library and then use a boolean function named KEY to replace all places in the program where you have a Keypress used. For a free copy of this in more detail, please request for same to M & R Enterprises, Box 61011, Sunnyvale, Ca 94088.



SERIAL HANDSHAKE MODIFICATION WITH TABS JUNE 1980

INTRODUCTION:

The High Speed Serial Interface card cannot run faster than 300 baud on most printers due to a lack of a printer busy line. This modification uses the existing data input line to sense if the printer is busy and inhibit output if necessary. DOS 3.2 is required to use this modification. This version lets TAB(X) work normally.

WARNING:

Damage to the High Speed Serial Interface card will not be covered by your warranty. If you aren't sure of the signal levels and pinout of your printer, find out or ask your Service Center to help you.

WIRING CHANGES:

First you must determine which wire your printer uses to indicate a printer busy or buffer full condition. Your printer's manual should contain this information or contact the manufacturer.

Examples:

IDS 125/225	pin 4
HEATH H-14	pin 4
TI-810	pin 11
SPINTERM	pin 19
COMPRINT	pin 20

The preferred place to do the wiring change is in the cable, but it can also be done at the Serial Card or the printer. Disconnect the wire between pin 2 of the printer and pin 2 on the Serial Card. Then connect the wire with the printer busy signal to the wire for pin 2 on the Serial Card.

SOFTWARE PATCH:

Next you must decide which slot the interface will go in and type in the software patch. The patch is customized for this slot number and won't work if used with a different configuration. The patch forces the computer to look to see if the printer is busy and wait if it is. Enter the patch using the values from the table for symbols in brackets, < >.

SLOT	1	2	3	4	5	6	7
A	90	A0	B0	C0	D0	E0	F0
B	C1	C2	C3	C4	C5	C6	C7
C	F9	FA	FB	FC	FD	FE	FF

Enter the monitor with CALL -155 and type

```
3A0:A9 <SLOT>
:20 95 FE
:A9 00
:20 ED FD
:A9 B5
:85 36
:A9 03
:85 37
:4C EA 03
:2C <A> C0
:30 FB
:20 07 <B>
:48
:AD <C> 05
:85 24
:68
:60
```

To check your typing, type

3A0L

and compare your listing to the one below for slot 1.

03A0-	A9 01	LDA	#\$01
03A2-	20 95 FE	JSR	\$FE95
03A5-	A9 00	LDA	#\$00
03A7-	20 ED FD	JSR	\$FDED
03AA-	A9 B5	LDA	#\$B5
03AC-	85 36	STA	\$36
03AE-	A9 03	LDA	#\$03
03B0-	85 37	STA	\$37
03B2-	4C EA 03	JMP	\$03EA
03B5-	2C 90 C0	BIT	\$C090
03B8-	30 FB	BMI	\$03B5
03BA-	20 07 C1	JSR	\$C107
03BD-	48	PHA	
03BE-	AD F9 05	LDA	\$05F9
03C1-	85 24	STA	\$85
03C3-	68	PLA	
03C4-	60	RTS	

Now return to basic with 3D0G.

SAVING THE PATCH TO DISK:

The patch must be in memory before the printer can be used at the higher speeds or with the TABbing feature. Save the patch by typing
BSAVE PATCH, A\$3A0, L\$25.

USING THE PRINTER:

The first time you want to use the printer you must load the patch and initialize the interface. From immediate mode type
BLOAD PATCH
CALL 928.

This may be done in a program, by entering
1000 PRINT D\$;"BLOAD PATCH": CALL 928
assuming that D\$ is a control-D.

If in the course of the program you need to turn off the printer, type
PR#0
or in a program enter
2000 PRINT D\$;"PR#0"

Then to reconnect the printer, all that is required is
CALL 938 or from a program
3000 CALL 938.

NOTES:

If the printer does not print after the CALL 928, it is probably sending the opposite polarity busy signal. The patch can be changed to recognize with
POKE 952,16.
If this doesn't work, have the printer checked.

The modification allows the speed, column width, and other variables to be changed with the POKEs in the manual.



APPLE POST - Mailing List Maintenance the Apple Way

Apple Post is a mailing list system designed for your Apple that allows you to enter, edit and store names, addresses and telephone numbers. When your Apple Computer System is attached to a compatible printer, Apple Post will also print mailing labels, address and telephone lists, and your personal zip code directory.

BENEFITS

Apple Post...

- Allows small businesses to take advantage of direct mail to customers, because it uses the power of the Apple computer to maintain customer lists and print mailing labels...
- Locates names that you wish to modify or which may have been misspelled during data entry, through its phonetic search routines...
- Provides you with the capability to perform demographic mailings quickly and easily, because it will print addresses by specified zip codes or other identifiers through its special search routines...
- Offers you the option of printing complete or specified portions of any list for your review or files through its select routine...
- Automatically prints mailing labels of completed or selected mailing lists through its labels routine.

APPLE POST - A CLOSER LOOK

An Apple Post list will hold approximately 500 name and address records on a single diskette, for use with a two-drive system. A maximum of 2,590 records, or five full diskettes, may be kept online at the same time with a six drive system. If you are going to be creating large lists, consider the possibility of breaking them up into smaller lists. For example, a subscription list for a monthly magazine could be broken into one list for the subscriptions that would expire in January, a second for those that would expire in February, etc. This would allow Apple Post to handle over 30,000 subscriptions with six drives, or 6,000 with two drives.

Learning the function of each Apple Post command is all it takes to begin using the Apple Post system. These commands appear in the VALID COMMAND LIST at the beginning of the program. A few are detailed here to give you a feeling for the program.

EXPLAIN may be used to display a brief three-line summary of any listed command in the system. It's especially helpful if you forget what a certain command will do.

The ENTER command permits you to add new entries to the Apple Post name and address list. After selecting this command, your screen will display the NEW ENTRY preformatted screen.

Most of the entries are self-explanatory. The ATTENTION line is a free line of 20 characters which is printed before the name line on mailing labels produced by the LABELS command. If the name line contains the name of a company, the attention line could contain the name of an individual or department within the company.

The SORT KEY is the line used by the Apple Post system to decide where on the diskette to file the entry. All entries are stored in dictionary order by sort key. The sort key line cannot be empty. When a name and address is first entered, a default reply will appear on the sort key line, for example, the person's last name. If you wish to change the sort key, type the new entry and the default reply will disappear as you type over it.

To "flag" or identify some names as belonging to special groups, Apple Post provides the UTILITY line. It will be referenced when you use the SELECT command to specify a group or groups for which you need to print a listing or mailing labels.

The EDIT command allows you to modify any line of a name and address record, except for the sort key line, or delete the entire entry.

If you are trying to locate a certain name and you're not sure of the way you spelled it during entry, you will use the FIND command. The spelling and spacing of the name need not be identical to what was entered; the Apple Post system will search for any entries that are phonetically similar to the one you have specified.

The ZIPFIND command selects name and address records by Zip Code and permits you to modify any line of the names and addresses in the list.

The Apple Post manual explains each command in detail.

TECHNICAL SPECIFICATIONS

Language: Written in Applesoft BASIC.

Function: Mailing List Maintenance

Maximum Size of an Online List: 2590 names

Field Sizes:

Name	25 characters
Street Address	25 characters
City & State	20 characters
Zip Code	9 characters
Attention Line	20 characters
Telephone Number	12 characters
Sort Key	10 characters
Utility Field	10 characters

Record Sizes:

Name & Address Files	155 bytes
Soundex Index File	17 bytes
Zip Code Index File	17 bytes

Disk Drive Requirements:

1 to 500 Names	2 drives
501 to 1000 Names	3 drives
1051 to 1570 Names	4 drives
1571 to 2105 Names	5 drives
2106 to 2590 Names	6 drives

THE APPLE POST PACKAGE

Order No. A2D0013

With your Apple Post order, you will receive:

- One (1) Apple Post diskette;
- Apple Post Mailing List System Manual.

SYSTEM CONFIGURATION

To use Apple Post, you will need:

- Apple II (with Applesoft II Firmware Card), or Apple II Plus, each with 32K memory;

or

- Apple II or Apple II Plus, each with the Apple Language System and 48K memory;
- Two (2) Apple Disk II drives, one with controller;
- a compatible printer* (optional);
- a printer controller card* (optional);

* Note: The Apple Computer System works with several printers and appropriate controller cards, including those specified below:

- Centronics
Card: Centronics Printer Interface Card (Apple Product A2B0007)
- Printronix
Card: Parallel Interface Card (Apple Product A2L0004)
- Qume Sprint 5, Diablo Hyterm, and NEC Spinwriter 5510R
Card: High Speed Serial Interface Card (Apple Product A2B0005) with P8-02 PROM



Upper/lower Case and Special Characters

This application note will describe a method of changing the Apple II computer so that control of upper and lower case is controlled with the shift key. This modification is for use with the Language System and includes a change to the BIOS so that upper and lower case can be used with the text editor supplied with the Language System as well as all user programs. In addition, all special characters like @, _, ~, ^, \, [,], ^, and { may be entered using the normal Apple II keyboard. This application note was prepared on an Apple II system equipped with this modification.

There are three parts to the complete modification. The hardware additions described here void the warranty on the Apple II computer.

The first part is to install a lower case display adapter like the Dan Paymar unit available from your local computer store or from several mail order houses. This allows the hardware to display the lower case characters once they are placed in the proper memory location.

The second part of the change requires that the shift key be wired to the third switch port on the same socket. If you are using SW2 for something else already, then you will have to change the program or lose your current use of SW2. SW2 was selected since generally there are two paddle controls with one switch on each one (SW0 and SW1) and because this hardware change was described in an early WOZPAK from Apple Puresound Program Library Exchange (A.P.F.L.E.). This modification is necessary so that the program change described later can tell if the shift key is pressed regardless of the condition of any other key.

If you do not already have this modification, this is how to install it. First, unplug the computer from the wall outlet and remove the keyboard cable plug from the main board. This electrically disconnects the keyboard from the rest of the computer. Next turn the computer over and remove the bottom cover from the computer. You now have two pieces; one piece is the light colored plastic piece containing the keyboard, and the other piece contains the main board and power supply. Next locate the area on the keyboard circuit board immediately under either one of the shift keys. you will notice that there are three short solder coated wires sticking out of the board. One of these wires has no foil leading away from it. Another one of the wires has a foil leading off and connecting to many other keys on the keyboard; this is the ground. The third one has a foil leading only to the other shift key and one or two other places; this is the one we want. Carefully solder a small diameter wire to the short wire protruding from the circuit board just identified. Next find the socket on the circuit board that has the cable plugged into it that goes to the main board. Locate an unused pin on that socket and carefully solder the other end of the wire to that pin. Pin 4 was used

in the prototype. This completes the modifications directly on the keyboard.

Now we have to get that connection over to the same paddle. Take the other assembly with the main board and power supply and remove the cable from the power supply to the main board. Then remove the main board from the metal bottom plate. Turn the main board over and carefully solder a wire from the keyboard connector socket, same pin number as on the keyboard circuit board, to pin 4 on the same paddle socket. This connects the keyboard shift key to SW2.

Now put everything back together in the exact reverse order that you took it apart. You now have a direct connection from the shift key to SW2. Plug everything back in and make sure that everything operates normally.

Now we will test the hardware modification. Key in the following BASIC program:

```
100 PRINT PEEK(-16285)
200 GOTO 100
```

Now run the program. The program looks at SW2 and prints out the value on the display screen. The value will be greater than or equal to 128 with the shift key not pressed, and will be less than 128 with the shift key pressed. Note that this has nothing at all to do with normal keyboard operation.

The third part of the modification is a change to the PASCAL BIOS to check the condition of the shift key under certain conditions and change the data actually read from the keyboard depending on the state of the shift key. The source for the patch is attached to this article. The patch program essentially reads an unchanged version of SYSTEM.APPLE from volume APPLE1, makes modifications in memory, and writes out the modified version on whatever disk is in unit number 4. The program is written to allow updating on a single drive system. The patch is set up to leave all characters in lower case unless the shift key is pressed. Special characters are obtained by pressing shift and control at the same time as follows:

```
Shift control P generates @
Shift control B generates {
Shift control G generates }
Shift control I generates "
Shift control J generates |
Shift control K generates [
Shift control L generates \
Shift control M generates ]
Shift control N generates ^
Shift control O generates _
Shift control R generates NULL
Shift control U generates `
```

Shift control Q is a shift lock and acts as a toggle. Start-up leaves the Apple keyboard as it came from the factory. Pressing shift control Q once makes it handle lower case and special characters. Press it again and it is back to factory operation.

Many times I have tried to get ahead of the computer only to find that I typed too quickly. This patch includes a feature that sounds a beep when the computer accepts a keystroke. The use of this feature will become obvious with use.

To apply the patch to the BIOS, enter the attached PASCAL program, compile it, and execute it. The program will ask you to put APPLE1 in device #4, read SYSTEM.APPLE, make the changes, ask you to put the target disk in device #4, and writes out a modified SYSTEM.APPLE. After the program is executed, you must re-boot the entire system by turning the power off and then on or by entering an 'H' from the command level.

That is all there is to it. This modification has been on my Apple II for several months and there have been absolutely no problems.

PROGRAM CHANGE:

```
VAR BUFFER: PACKED ARRAY [0..31,0..511] OF 0..255;
    F:FILE;
    I:INTEGER;
```

```
FUNCTION HEX(X:CHAR):INTEGER;
BEGIN
    IF X IN ['0'..'9'] THEN
        HEX:=ORD(X)-48
    ELSE
        IF X IN ['A'..'F'] THEN
            HEX:=ORD(X)-55
        ELSE
            HEX:=32767
        END;
    END; (*HEX*)
```

```
PROCEDURE CB(LOC,DAT:STRING);
VAR L,LN,L1,L2,L3,L4,REC,DISP,D1:INTEGER;
BEGIN
    L1:=HEX(LOC[1]);
    L2:=HEX(LOC[2]);
    L3:=HEX(LOC[3]);
    L4:=HEX(LOC[4]);
    REC:=(L1-13)*8+L2 DIV 2;
    DISP:=(L2 MOD 2)*256+L3*16+L4;
    LN:=LENGTH(DAT) DIV 2;
    WRITELN(LOC,',',',',DAT,',',',',LN);
    FOR L:=1 TO LN DO BEGIN
        D1:=HEX(DAT[L-1])*16+HEX(DAT[L]);
        WRITELN(REC,',',',',DISP,',',',',D1);
        BUFFER[REC,DISP]:=D1;
        DISP:=DISP+1;
        IF DISP=512 THEN BEGIN
            REC:=REC+1;
            DISP:=0;
        END;
    END;
    END; (*CB*)
```

PROCEDURE READIN;

VAR XXX: CHAR;

BEGIN

WRITELN('INSERT "APPLE1", AND PRESS "RETURN"');

READ(XXX);

RESET(F, 'APPLE1:SYSTEM.APPLE');

I:=BLOCKREAD(F,BUFFER,32);

CLOSE(F);

END;

PROCEDURE WRITEOUT;

VAR XXX: CHAR;

BEGIN

WRITELN('INSERT TARGET DISKETTE AND PRESS "RETURN"');

READ(XXX);

RESET(F, '#4:SYSTEM.APPLE');

I:=BLOCKWRITE(F,BUFFER,32);

CLOSE(F);

END;

BEGIN

READIN;

CB('D8E8','EAEAEAEA');(*NO-OP CONVERT TO UPPER CASE*)

CB('D69F','297F');(*AND #57F RESET HIGH ORDER BIT*)

CB('D6A1','20BEDA');(*JSR \$DABE PATCH*)

CB('DABE','8D10C0');(*RESET KBD STROBE*)

CB('DAC1','48');(*PHA SAVE INPUT DATA*)

CB('DAC2','A006');(*LDY #\$06*)

CB('DAC4','A200');(*LDX #\$00*)

CB('DAC6','CA');(*DEX*)

CB('DAC7','D0FD');(*BNE \$DAC6*)

CB('DAC9','AD30C0');(*LDA \$C030 SPEAKER*)

CB('DACC','88');(*DEY*)

CB('DACD','D0F5');(*BNE \$DAC4*)

CB('DACF','68');(*PLA RESTORE INPUT DATA*)

CB('DAD0','C911');(*CMF #11 TEST FOR CONTROL Q*)

CB('DAD2','D012');(*BNE \$DAE6 BRANCH IF NOT CONTROL Q*)

CB('DAD4','2C63C0');(*BIT \$C063 TEST SHIFT*)

CB('DAD7','300D');(*BNE \$DAE6 BRANCH IF NOT SHIFT*)

CB('DAD9','A980');(*LDA #\$80*)

CB('DADB','4DECDA');(*EOR \$DAEC TOGGLE THE SWITCH*)

CB('DADE','8DECDA');(*STA \$DAEC STORE IT BACK*)

CB('DAE1','6868');(*PLA PLA REMOVE RETURN ADDRESS*)

CB('DAE3','4C25D7');(*JMP \$D725 JUMP BACK, NO DATA*)

CB('DAE6','2CECDA');(*BIT \$DAEC TEST THE SWITCH*)

CB('DAE9','1002');(*BPL \$DAEA BRANCH IF NOT LOCKED*)

CB('DAEB','60');(*RTS RETURN WITH ORIGINAL DATA*)

CB('DAEC','80');(*SWITCH LOCATION, FACTORY SETTING*)

CB('DAED','C941');(*CMP #41 COMPARE FOR 'A'*)

CB('DAEF','900C');(*BCC \$DAFD BRANCH IF LT 'A'*)

CB('DAF1','C95B');(*CMP #5B COMPARE FOR Z+1*)

CB('DAF3','B01F');(*BCS \$DB14 BRANCH IF GREATER THAN 'Z'*)

CB('DAF5','2C63C0');(*BIT \$C063 TEST SHIFT KEY*)

CB('DAF8','1002');(*BPL \$DAFC BRANCH IF SHIFT PRESSED*)

```

CB('DAFA','6920'); (*ADC ##20 CONVERT TO LOWER CASE*)
CB('DAFC','60'); (*RTS RETURN TO CALLER*)
CB('DAFD','C940'); (*CMP ##40 COMPARE FOR @*)
CB('DAFF','D003'); (*BNE $DB04 BRANCH IF NOT @*)
CB('DB01','A950'); (*LDA ##50 SET CAP F*)
CB('DB03','60'); (*RTS RETURN WITH CAP F*)
CB('DB04','C920'); (*CMP ##20 COMPARE FOR SPACE*)
CB('DB06','9001'); (*BCC $DB09 BRANCH IF CONTROL CHAR*)
CB('DB08','60'); (*RTS RETURN IF NOT CONTROL CHAR*)
CB('DB09','2C63C0'); (*BIT $C063 TEST SHIFT KEY*)
CB('DB0C','1001'); (*BPL $DB0F BRANCH IF SHIFT*);
CB('DB0E','60'); (*RTS RETURN IF CONTROL ONLY*)
CB('DB0F','AA'); (*TAX TRANSFER A TO X*)
CB('DB10','BD23DB'); (*LDA $DB23,X LOAD NEW VALUE FROM TABLE*)
CB('DB13','60'); (*RTS RETURN WITH NEW CHAR*)
CB('DB14','C95E'); (*CMP ##5E COMPARE FOR ^*)
CB('DB16','D003'); (*BNE $DB1B BRANCH IF NOT ^*)
CB('DB18','A94E'); (*LDA ##4E CHANGE TO CAP N*)
CB('DB1A','60'); (*RTS RETURN WITH CAP N*)
CB('DB1B','C95D'); (*CMP ##5D COMPARE FOR ]*)
CB('DB1D','D003'); (*BNE $DB22 BRANCH IF NOT*)
CB('DB1F','A94D'); (*LDA ##4D CHANGE TO CAP M*)
CB('DB21','60'); (*RTS RETURN WITH CAP M*)
CB('DB22','60'); (*RTS RETURN WITH WHATEVER*)
CB('DB23','40017B0304051C7D1D'); (*TABLE FOR CHANGE...*)
CB('DB2C','7E7C5B5C5D5E5F1011'); (*....OF SHIFT PLUS CNTL....*)
CB('DB35','00131E601F17187F1A'); (*....TO SPECIAL ....*)
CB('DB3E','1B1C5D5E1F'); (*....CHARACTERS*)
WRITEOUT;
END.

```



INTERNATIONAL
APPLE CORE

TM

P. O. BOX 976, DALY CITY, CALIFORNIA 94017 USA

APNOTE

```
(*****)  
(* *)  
(* LINEFEED *)  
(* *)  
(* INITIALIZE BIOS TO FILTER OUT *)  
(* ANY LINEFEEDS SENT TO PRINTER. *)  
(* *)  
(* UCSD SYSTEM OF 23 FEB 79 FOR *)  
(* APPLE II HAS A LINE-FEED FLAG *)  
(* AT LOCATION BFOF HEX. *)  
(* IF THIS FLAG IS SET TO 255, *)  
(* LINE-FEEDS WILL BE SUPPRESSED. *)  
(* IF IT IS SET TO 0 (DEFAULT), *)  
(* THEN LINE-FEEDS WILL BE PASSED. *)  
(* *)  
(*****)
```

PROGRAM LINEFEED;

```
TYPE PA=PACKED ARRAY[0..1] OF 0..255;  
    TWOFACE=RECORD CASE BOOLEAN OF  
        TRUE: (INT:INTEGER);  
        FALSE: (PTR:^PA);  
    END;
```

VAR CHEAT:TWOFACE;

```
BEGIN  
    CHEAT.INT:=-16625; (* BFOF HEX *)  
    CHEAT.PTR^[0]:=255; (* SET FLAG *)  
END.
```




**INTERNATIONAL
APPLE CORE**

TM

P. O. BOX 976, DALY CITY, CALIFORNIA 94017 USA

APNOTE

```

PROGRAM TAKE280;

(*****
(*                                                                    *)
(*  ACCEPTS 280 BLOCKS FROM REMIN AND WRITES TO #5                    *)
(*                                                                    *)
(******
CONST REMIN=7;
    REMOUT=8;

VAR BUFFER: PACKED ARRAY [0..2048] OF INTEGER;
    I,SUM,BLOCK,TRACK: INTEGER;
    REPLY: PACKED ARRAY [0..1] OF CHAR;
    CH: CHAR;

BEGIN (* PROGRAM *)
    REPLY[0]:=^A^;
    REPEAT (* FOR EACH DISK *)
        Writeln(^PUT BLANK DISK IN #5 AND PRES RETURN^);
        ReadLn;
        UnitClear(REMIN);
        Writeln(^FIRE WHEN READY..^);
        SUM:=0;
        FOR TRACK:=0 TO 34 DO
            BEGIN
                BLOCK:=8*TRACK;
                UnitRead(REMIN,BUFFER,4098,0,12);
                FOR I:=0 TO 2047 DO SUM:=SUM+BUFFER[I];
                IF BUFFER[2048]=SUM THEN
                    BEGIN
                        UnitWrite(5,BUFFER,4096,BLOCK);
                        Write(^.);
                        UnitWrite(REMOUT,REPLY,1);  (* SAY READY FOR MORE *)
                    END
                ELSE
                    BEGIN
                        Write(^CHECKSUM ERROR^);
                        Exit(PROGRAM);
                    END;
            END;
        Writeln;
        Write(^COPY ANOTHER DISK ? ^);
        Read(CH);
        Writeln;
        UNTIL CH<>^Y^;
        Writeln;Writeln(^THAT^S ALL FOLKS...^);
    END.

```



PROGRAM GETREM;

```
(*****)  
(* *)  
(* READS FROM REMIN AND WRITES TO THE DISK *)  
(* *)  
(*****)
```

CONST REMIN=7;

```
VAR BUFFER: PACKED ARRAY [0..16383] OF 0..255;  
    BLOCK,NBLOCKS,BYTES,UNITNUM: INTEGER;  
    CH: CHAR;  
    F: FILE;  
    FNAME: STRING[30];
```

PROCEDURE WRITEFILE;

```
BEGIN  
    (*$I-*)  
    REPEAT  
        WRITELN;  
        WRITE('TO WHAT FILE ? ');  
        READLN(FNAME);  
        REWRITE(F,FNAME);  
    UNTIL IORESULT=0;  
    (*$I+*)  
    IF BLOCKWRITE(F,BUFFER,NBLOCKS)<>NBLOCKS  
        THEN WRITELN('FILE WRITE ERROR.');
```

```
    CLOSE(F,LOCK);
```

END;

PROCEDURE WRITEUNIT;

```
BEGIN  
    WRITELN;  
    WRITE('WHICH UNIT NUMBER ? ');  
    READLN(UNITNUM);  
    WRITE('STARTING BLOCK ? ');  
    READLN(BLOCK);  
    UNITWRITE(UNITNUM,BUFFER,BYTES,BLOCK,12);  
END;
```

```

BEGIN
  REPEAT (* FOR EACH TRANSFER *)
    PAGE(OUTPUT);
    REPEAT
      WRITE('READ HOW MANY BLOCKS FROM REMIN ? ');
      READLN(NBLOCKS);
    UNTIL NBLOCKS<=32;
    BYTES:=NBLOCKS*512;
    WRITELN('FIRE WHEN READY...');
    UNITCLEAR(REMIN);
    UNITREAD(REMIN,BUFFER,BYTES,0,12);
    WRITELN('ALL ',NBLOCKS,' RECEIVED. ');
    REPEAT
      WRITE('F(ILE, U(NIT, Q(UIT ');
      READ(CH);
    UNTIL CH IN ['F','U','Q'];
    CASE CH OF
      'F': WRITEFILE;
      'U': '.....',
      'Q': EXIT(PROGRAM);
    END;
    WRITELN;
    WRITE('DONE WRITING. GET SOME MORE ? ');
    REPEAT
      READ(CH);
    UNTIL CH IN ['Y','N'];
  END.

```



P. O. BOX 976, DALY CITY, CALIFORNIA 94017 USA

```
PROGRAM TRANSFER;
(*****)
(*                                           *)
(* SENDS AND RECEIVES FILES OR WHOLE VOLUMES OVER SERIAL LINE *)
(*                                           *)
(*****)
CONST VERSION='A.1'; (* CHANGE WITH EACH REVISION *)
    REMIN=7;
    REMOUT=8;
    CONSOLE=1;

TYPE SETOFCHAR=SET OF CHAR;
    BUFTYPE=RECORD
        DATA: PACKED ARRAY[0..511] OF 0..255;
        CKSUM: INTEGER;
    END;

VAR BUFFER:BUFTYPE;
    CH: CHAR;
    GOODKEY: BOOLEAN;
    LEADIN,ERASEOL,ERASEOS: CHAR;

FUNCTION SUM512(VAR BUFFER:BUFTYPE): INTEGER;
EXTERNAL;

PROCEDURE SEND;
FORWARD;

PROCEDURE RECEIVE;
FORWARD;

PROCEDURE GETCRTINFO;
VAR BUFFER: PACKED ARRAY[0..511] OF CHAR;
    I: INTEGER;
    F: FILE;
BEGIN
    RESET(F,'*SYSTEM.MISCINFO');
    I:=BLOCKREAD(F,BUFFER,1);
    CLOSE(F);
    LEADIN:=BUFFER[62];
    ERASEOS:=BUFFER[65];
    ERASEOS:=BUFFER[64];
END;

PROCEDURE CLEARSCREEN;
BEGIN
    GOTOXY(0,0);
    WRITE(LEADIN,ERASEOS);
END;

PROCEDURE CLEARLINE;
BEGIN
    WRITE(LEADIN,ERASEOL);
```

This application note has been provided by an Apple Computer user. The International Apple Core does not guarantee the accuracy of this information in any way and cautions that modifications may void a manufacturer's warranty. The International Apple Core is a non-profit organization unaffiliated with Apple Computer, Inc. or any other manufacturer. Apple II, Applosoft, and Apple Computer are trademarks of Apple Computer, Inc.

```

END;

PROCEDURE PROMPTAT(Y: INTEGER; S: STRING);
BEGIN
    GOTOXY(0,Y);
    WRITE(S);
    CLEARLINE;
END;

FUNCTION GETCHAR(OKSET: SETOFCHAR): CHAR;
VAR CH: CHAR;
    GOOD: BOOLEAN;
BEGIN
    REPEAT
        READ(KEYBOARD,CH);
        GOOD:=CH IN OKSET;
        IF GOOD THEN WRITE(CH) ELSE WRITE(CHR(7));
    UNTIL GOOD;
    GETCHAR:=CH;
END;

PROCEDURE SENDREPLY(S: STRING);
BEGIN
    UNITWRITE(REMOUT,S,5,0,12);
END;

FUNCTION GETBLOCK:BOOLEAN;
(* RETURNS TRUE IF CHECKSUM MATCHED *)
BEGIN
    UNITREAD(REMIN,BUFFER,514,0,12); (* GET DATA AND CHECKSUM *)
    GETBLOCK:=BUFFER.CKSUM=SUM512(BUFFER);
END;

PROCEDURE SENDBLOCK;
(* SEND BLOCK + CHECKSUM AND REPEAT UNTIL REPLY 'GOOD' *)
VAR REPLY: STRING[4];
    I: INTEGER;
BEGIN
    REPEAT
        BUFFER.CKSUM:=SUM512(BUFFER);
        UNITWRITE(REMOUT,BUFFER,514,0,12); (* SEND DATA AND CHECKSUM *)
        UNITREAD(REMIN,REPLY,5,0,12);
    UNTIL REPLY='GOOD';
    FOR I:=1 TO 100 DO;
END;

PROCEDURE GETVOLUME;
(* GET WHOLE VOLUME, INCLUDING DIRECTORY *)
VAR BLOCK,NBLOCKS,DRIVE,ERRORS,TOTALSUM,SUM: INTEGER;
BEGIN
    (*$I-*)
    REPEAT
        PROMPTAT(4,'RECEIVE HOW MANY BLOCKS ? ');
        READLN(NBLOCKS);
    UNTIL IORESULT=0;

```

```

IF NBLOCKS=0 THEN EXIT(RECEIVE);
REPEAT
  PROMPTAT(6,'WRITE TO WHICH DRIVE ? ');
  READLN(DRIVE);
UNTIL (IORESULT=0) AND (DRIVE IN [4,5,9,10,11,12]);
(*$I+*)
WRITELN;
WRITE('PUT BLANK DISK IN #',DRIVE,' AND PRESS RETURN. ');
UNITCLEAR(CONSOLE); (* FLUSH TYPE-AHEAD *)
READLN;
WRITELN;
WRITELN('FIRE WHEN READY... ');
UNITCLEAR(REMIN);
WRITELN;
TOTALSUM:=0;
FOR BLOCK:=0 TO NBLOCKS-1 DO
  BEGIN
    ERRORS:=0;
    REPEAT
      IF GETBLOCK THEN
        BEGIN
          UNITWRITE(DRIVE,BUFFER,512,BLOCK,12);
          WRITE('. ');
          IF (BLOCK MOD 35)=34 THEN WRITELN;
          SENDREPLY('GOOD');
        END
      ELSE
        BEGIN
          ERRORS:=ERRORS+1;
          WRITE(ERRORS);
          SENDREPLY('BAD');
          IF ERRORS>3 THEN
            BEGIN
              WRITELN;
              WRITELN(CHR(7),'MORE THAN 3 CHECKSUM ERRORS');
              WRITELN('PROGRAM ABORTED');
              EXIT(PROGRAM);
            END;
        END;
    UNTIL ERRORS=0; (* OR EXIT WITH ERRORS > 3 *)
    TOTALSUM:=TOTALSUM+BUFFER.CKSUM;
  END;
WRITELN;
WRITELN('ALL ',NBLOCKS,' BLOCKS WRITTEN TO #',DRIVE);
SUM:=0;
FOR BLOCK:=0 TO NBLOCKS-1 DO
  BEGIN
    UNITREAD(DRIVE,BUFFER,512,BLOCK,12);
    SUM:=SUM+SUM512(BUFFER);
  END;
IF SUM=TOTALSUM THEN WRITELN('MASTER CHECKSUM MATCHES');
ELSE WRITELN(CHR(7),'ERROR: MASTER CHECKSUM DOESN'T MATCH');
END; (* GETVOLUME *)

PROCEDURE SENDVOLUME;

```

```

(* SEND WHOLE VOLUME, INCLUDING DIRECTORY *)
VAR BLOCK,NBLOCKS,DRIVE: INTEGER;
BEGIN
  (*$I-*)
  REPEAT
    PROMPTAT(4,'SEND HOW MANY BLOCKS ? ');
    READLN(NBLOCKS);
  UNTIL IORESULT=0;
  IF NBLOCKS=0 THEN EXIT(SEND);
  REPEAT
    PROMPTAT(6,'READ FROM WHICH DRIVE ? ');
    READLN(DRIVE);
  UNTIL (IORESULT=0) AND (DRIVE IN [4,5,9,10,11,12]);
  (*$I+*)
  WRITELN;
  WRITE('PUT SOURCE IN #',DRIVE,' AND PRESS RETURN');
  UNITCLEAR(CONSOLE); (* FLUSH TYPE-AHEAD *)
  UNITCLEAR(REMIN);
  READLN;
  FOR BLOCK:=0 TO NBLOCKS-1 DO
    BEGIN
      UNITREAD(DRIVE,BUFFER,512,BLOCK,12);
      SENDBLOCK;
      WRITE('.');
      IF (BLOCK MOD 35)=34 THEN WRITELN;
    END;
  WRITELN;
  WRITELN;
  WRITELN('ALL ',NBLOCKS,'BLOCKS SENT FROM #',DRIVE);
END; (* SENDVOLUME *)

PROCEDURE GETFILE;
VAR MESSAGE: STRING[4];
    I,BLOCK,ERRORS: INTEGER;
    F: FILE;
    FNAME: STRING[30];
BEGIN
  (*$I-*)
  REPEAT
    PROMPTAT(4,'RECEIVE WHAT TEXTFILE ? ');
    READLN(FNAME);
    IF LENGTH(FNAME)=0 THEN EXIT(RECEIVE);
    IF POS('.TEXT',FNAME)=0 THEN FNAME:=CONCAT(FNAME,'.TEXT');
    REWRITE(F,FNAME);
  UNTIL IORESULT=0;
  (*$I+*)
  UNITCLEAR(REMIN);
  WRITELN;
  WRITELN('FIRE WHEN READY...');
  WRITELN;
  BLOCK:=0;
  REPEAT
    UNITREAD(REMIN,MESSAGE,5);
    IF MESSAGE<>'DONE' THEN
      BEGIN

```

```

    IF GETBLOCK THEN
    BEGIN
        I:=BLOCKWRITE(F,BUFFER,1,BLOCK);
        BLOCK:=BLOCK+1;
        WRITE(' ');
        IF (BLOCK MOD 35)=34 THEN WRITELN;
        SENDREPLY('GOOD');
    END
ELSE
    BEGIN
        ERRORS:=ERRORS+1;
        WRITE(ERRORS);
        SENDREPLY('BAD');
        IF ERRORS>3 THEN
        BEGIN
            WRITELN;
            WRITELN(CHR(7),'MORE THAN 3 ERRORS. ');
            WRITELN('PROGRAM ABORTED');
            EXIT(PROGRAM);
        END;
    END;
END;
UNTIL MESSAGE='DONE';
WRITELN;
WRITELN;
WRITELN(BLOCK,' BLOCKS WRITTEN TO ',FNAME);
CLOSE(F,LOCK);
END; (* GETFILE *)

PROCEDURE SENDFILE;
VAR I,BLOCK: INTEGER;
    F: FILE;
    FNAME: STRING[30];
BEGIN
    UNITCLEAR(REMIN);
    WRITELN;
    (*$?-*)
    REPEAT
        PROMPTAT(4,'SEND WHAT TEXTFILE ? ');
        READLN(FNAME);
        IF LENGTH(FNAME)=0 THEN EXIT(SEND);
        RESET(F,FNAME);
        IF IORESULT<>0 THEN
        BEGIN
            FNAME:=CONCAT(FNAME,'.TEXT');
            RESET(F,FNAME);
        END;
    UNTIL IORESULT=0;
    (*$I+*)
    WRITELN;
    WRITE('PRESS RETURN TO SEND ',FNAME,' ');
    UNITCLEAR(CONSOLE);
    READLN;
    WRITELN;
    UNITCLEAR(REMIN);

```



```

BLOCK:=0;
WHILE NOT EOF(F) DO
  BEGIN
    I:=BLOCKREAD(F,BUFFER,1,BLOCK);
    SENDREPLY('SYNC');
    FOR I:=1 TO 100 DO;
      SENDBLOCK;
      WRITE(',');
      IF (BLOCK MOD 35)=34 then writeln;      BLOCK:=BLOCK+1;
    END;
    CLOSE(F);
    FOR I:=1 TO 100 DO;
      SENDREPLY('DONE');
    WRITELN;
    WRITELN(BLOCK,' BLOCKS SENT FROM ',FNAME);
  END;

PROCEDURE SEND;
VAR CH: CHAR;
BEGIN
  PROMPTAT(2,'SEND F(ILE OR V(OLUME ? ');
  CH:=GETCHAR(['F','f','V','v']);
  CASE CH OF
    'F','f': SENDFILE;
    'V','v': SENDVOLUME;
  END;
END; (* SEND *)

PROCEDURE RECEIVE;
VAR CH: CHAR;
BEGIN
  PROMPTAT(2,'RECEIVE F(ILE OR V(OLUME ? ');
  CH:=GETCHAR(['F','f','V','v']);
  CASE CH OF
    'F','f': GETFILE;
    'V','v': GETVOLUME;
  END;
END; (* RECEIVE *)

BEGIN (* MAIN PROGRAM *)
  GETCRTINFO;
  CLEARSCREEN;
  REPEAT
    PROMPTAT(0,CONCAT('>TRANSFER: S(END, R(ECEIVE, Q(UIT ',VERSION));
    CASE CH OF
      'S','s': SEND;
      'R','r': RECEIVE;
    END;
  UNTIL CH IN ['Q','q'];
  WRITELN;
  WRITELN('THAT''S ALL FOLKS...');
END.

```

```

        .MACRO POP
        PLA
        STA Z1
        PLA
        STA Z1+1
        .ENDM

        .MACRO PUSH
        LDA Z1+1
        PHA
        LDA Z1
        PHA
        .ENDM

        .FUNC SUM512,1
;-----
;  FUNCTION SUM512(VAR BUFFER): INTEGER;
;
RETURN   .EQU 0
BUFFER   .EQU 2
SUM       .EQU 4

        POP RETURN
        PLA
        PLA
        PLA
        PLA
        POP BUFFER
        LDY #0
        STY SUM
        STY SUM+1
LOOP1    CLC
        LDA SUM
        ADC @BUFFER,Y
        STA SUM
        BCC NEXT1
        INC SUM+1
LOOP2    CLC
        LDA SUM
        ADC @BUFFER,Y
        STA SUM
        BCC NEXT2
        INC SUM+1
NEXT2    INY
        BNE LOOP2
        PUSH SUM
        PUSH RETURN
        RTS
        .END

```



PROGRAM FOREIGN;

```
(*****)  
(*  
(* THIS PROGRAM ALSO USES THE ASSEMBLY ROUTINE "GETTEXT" *)  
(*  
(*****)
```

```
CONST BUFLen=1600; (* BUFFER SIZE IN BYTES *)  
LINELEN=132; (* MAX LINE LENGTH ALLOWED IN OUTPUT *)
```

```
VAR BUFFER: PACKED ARRAY [0..BUFLen] OF CHAR;  
NLines, I, NBytes, SKIPBUFS, SKIPBYTES, BUFPTR: INTEGER;  
S: STRING[255];  
FNAME: STRING[30];  
F: TEXT;
```

```
FUNCTION GETTEXT(VAR BUFFER, MAXBYTES, SKIPBYTES: INTEGER): INTEGER;  
EXTERNAL;
```

```
PROCEDURE GOODBYE;  
BEGIN  
  WRITELN; WRITELN; WRITELN('THAT'S ALL FOLKS...');  
  EXIT(PROGRAM);  
END;
```

```
FUNCTION GETSTRING: BOOLEAN;  
(* GET STRING S FROM BUFFER. RETURN FALSE IF NO MORE IN BUFFER *)  
BEGIN
```

```
  IF BUFPTR >= NBytes THEN GETSTRING:=FALSE  
  ELSE  
    BEGIN  
      I:=SCAN(NBytes-BUFPTR, =CHR(13), BUFFER[BUFPTR]);  
      GETSTRING:=TRUE;  
      IF I > LINELEN THEN (* GET PORTION OF LONG LINE *)  
        BEGIN  
          (*$R-*)  
          S[0]:=CHR(LINELEN);  
          MOVELEFT(BUFFER[BUFPTR], S[1], LINELEN);  
          (*$R+*)  
          BUFPTR:=BUFPTR+LINELEN;  
          I:=I-LINELEN;  
        END  
      ELSE
```

```
        BEGIN  
          (*$R-*)  
          S[0]:=CHR(I);  
          MOVELEFT(BUFFER[BUFPTR], S[1], I);  
          (*$R+*)  
          BUFPTR:=BUFPTR+I+1;  
        END;  
      END;  
    END;
```

```
  END;  
END;
```

```

BEGIN
  REPEAT
    (*$I-*)
    REPEAT
      WRITELN;WRITELN;
      WRITE('OUTPUT FILE NAME ? ');
      READLN(FNAME);
      IF FNAME='' THEN GOODBYE;
      IF POS('.TEXT',FNAME)=0 THEN FNAME:=CONCAT(FNAME,'.TEXT');
      REWRITE(F,FNAME);
    UNTIL IORESULT=0;
    (*$I+*)
    WRITELN('SKIP HOW MANY BYTES OF EACH LINE ? ');
    WRITE('(22 FOR TEDOS LISTINGS.) ');
    READLN(SKIPBYTES);
    WRITE('SKIP HOW MANY BUFFERS ? ');
    READLN(SKIPBUFS);
    NLINES:=0;
    WRITELN('FIRE WHEN READY...');
    WRITELN('(PRESS RETURN WHEN TRANSFER DONE.)');

    (*****
    (*
    (*  USE GETTEXT TO SKIP OVER THE FIRST SKIPBUFS BUFFERS.
    (*  THIS LETS YOU BREAK UP BIG FILES INTO SMALLER ONES BY
    (*  SENDING SEVERAL TIMES AND CAPTURING DIFFERENT PORTIONS
    (*  OF THE TEXT.
    (*
    (*****

    FOR I:=1 TO SKIPBUFS DO
      NBYTES:=GETTEXT(BUFFER,BUFLEN-2000,SKIPBYTES);

      (*****
      (*
      (*  DONE SKIPPING, GO GET THE DESIRED CHARACTERS
      (*  STOP WHEN RETURN PRESSED OR BUFFER FULL
      (*
      (*****

      NBYTES:=GETTEXT(BUFFER,BUFLEN,SKIPBYTES);
      WRITELN(NBYTES,' BYTES RECEIVED');
      BUFPTR:=0;
      WHILE GETSTRING DO
        BEGIN
          WRITELN(F,S);
          NLINES:=NLINES+1;
        END;
      CLOSE(F,LOCK);
      WRITELN(NLINES,' LINES WRITTEN TO ',FNAME);
    UNTIL FALSE;
  END.

```

```

;-----
;
; THIS ROUTINE IS USED IN THE PROGRAM "FOREIGN"
;
;
; .MACRO POP
; PLA
; STA Z1
; PLA
; STA Z1+1
; ENDM
;
; .MACRO PUSH
; LDA Z1+1
; PHA
; LDA Z1
; PHA
; ENDM
;
; .FUNC GETTEXT,3
;-----
;
; FUNCTION GETTEXT(VAR BUFFER; MAXBYTES,SKIPBYTES: INTEGER): INTEGER;
;
; RETURNS NUMBER OF BYTES READ FROM COM CARD SLOT 2 BEFORE CR
; PRESSED ON APPLE
;
; RETURN .EQU 0
; MAXBYTES .EQU 2
; BUFFER .EQU 4
; NBYTES .EQU 6
; CHARS .EQU 8
; SKBYTES .EQU 0A
; SKCOUNT .EQU 0C
; STATUS .EQU 0C0AE ;UART STATUS REGISTER, COM CARD SLOT 2 ONLY
; DATA .EQU 0C0AF
; KEYBOARD .EQU 0C000 ;APPLE KEYBOARD
; KEYRESET .EQU 0C010
;
; POP RETURN ;SAVE RETURN ADDRESS
; PLA ;DISCARD 4 BYTES STACK BIAS
; PLA
; PLA
; PLA
; POP SKBYTES ;HOW MANY BYTES TO SKIP AFTER CR
; POP MAXBYTES ;GET MAX BUFFER LENGTH
; POP BUFFER ;GET ADDRESS OF BUFFER
; LDA #0
; STA NBYTES
; STA NBYTES+1 ;NBYTES=0
; STA CHARS
; STA SKCOUNT
; LDA KEYBOARD
; CMP #141 ;HAS CR BEEN PRESSED?
; BEQ DONE ;IF YES, QUIT

```

```

LDA STATUS      ;TEST UART STATUS
ROR A
BCC WAIT        ;WAIT FOR A CHARACTER
LDA DATA       ;GET THE DATA FROM UART
AND #7F         ;MASK OFF BIT 7
CMP #32         ;IS IT A CONTROL CHARACTER?
BCS STORIT      ;NO, STORE IN BUFFER
CMP #13         ;IS IT A CR?
BEQ CR
CMP #9          ;IS IT A TAB?
BNE WAIT        ;NO, IGNORE IT
SPACES LDA #32   ;YES, STORE SPACES UNTIL MULT OF 8
JSR STOREONE
BEQ DONE        ;BUFFER FULL
LDA CHARS
AND #7
BNE SPACES
BEQ WAIT
CR LDY #0
STY CHARS       ;RESET # CHARS THIS LINE
LDY SKBYTES
STY SKCOUNT    ;RESET COUNT OF BYTES TO SKIP EACH LINE
JSR DOIT        ;STORE IN BUFFER
BEQ DONE        ;BUFFER FULL?
BNE WAIT        ;WAIT FOR MORE.
STORIT JSR STOREONE ;PUT CHAR IN BUFFER
BEQ DONE        ;BUFFER FULL
BNE WAIT        ;GO FOR ANOTHER
STOREONE INC CHARS ;BUMP LOGICAL COUNT CHARS THIS LINE
LDY SKCOUNT    ;ARE WE STILL SKIPPING CHARS THIS LINE
BEQ DOIT        ;NO, STORE IN BUFFER
DEC SKCOUNT    ;YES, DEC COUNTER
LDA #1          ;CLEAR ZERO FLAG (BUFFER NOT FULL)
RTS
DOIT LDY #0
STA (BUFFER),Y  ;PUT CHARACTER IN BUFFER (Y=0)
INC BUFFER
BNE SKIP1
INC BUFFER+1    ;BUMP BUFFER POINTER (16 BIT)
SKIP1 INC NBYTES
BNE SKIP2
INC NBYTES+1
SKIP2 SEC
LDA MAXBYTES
SBC #1
STA MAXBYTES
STA MAXBYTES+1
SBC #0
STA MAXBYTES+1
ORA MAXBYTES    ;SET ZERO FLAG IF BUFFER FULL
RTS
DONE PUSH NBYTES
PUSH RETURN
RTS             ;BACK TO PASCAL
.END

```



PROGRAM MAIN;

VAR F: TEXT;

(* THE FOLLOWING PROCEDURE SETS THE COMMUNICATIONS CARD IN A *)
(* GIVEN SLOT TO 110 BAUD, 8 BITS, NO PARITY. SEE THE COM *)
(* CARD MANUAL. THE COM CARD WILL STAY AT THAT BAUD RATE *)
(* UNTIL SET AGAIN. *)

PROCEDURE SET110BAUD;

CONST SLOT=2;

CODE=82; (* SEE COM CARD MANUAL *)

TYPE PA=PACKED ARRAY [0..1] OF 0..255;

TRIX=RECORD CASE BOOLEAN OF

TRUE: (INT:INTEGER);

FALSE: (PTR:~PA);

END;

VAR CARD: TRIX;

REM: TEXT;

X: STRING[99];

BEGIN

CARD.INT:=-16242+SLOT*16;

CARD.PTR^[0]:=3; (* RESET 6850 *)

CARD.PTR^[0]:=CODE; (* SET 110 BAUD *)

END;

BEGIN (* MAIN PROGRAM *)

REWRITE(F,~REMOUT:~);

WRITELN(F,~THIS IS AT 300 BAUD~);

SET110BAUD;

WRITELN(F,~THIS IS AT 110 BAUD~);

END.



THE PRELIMINARY APPLE PASCAL GUIDE TO INTERFACING FOREIGN HARDWARE 10 DEC 1979 (minor revision)

This document is intended to direct users of the APPLE II who are interfacing to their machine hardware other than Apple peripheral cards. Its primary target is the community of peripheral card manufacturers who have developed a product for the Apple II under BASIC, and wish to develop an end-user Pascal interface. It is assumed that the reader is familiar with Apple II hardware and is an experienced programmer at the machine level, and has programmed in Pascal.

How Apple Pascal looks at the outside world

Currently, Apple Pascal is capable of recognizing the presence of an Apple (brand) peripheral card in slots 1 through 7, which it does at boot time by scanning the ROM in the slot address space. The purpose of this scan is (a) to determine if any card exists in the slot, since an open slot yields a random response; and (b) to find out what kind of Apple card, if any, is there, since the four types of Apple cards (printer, com, disk, and serial) have distinct values at byte locations Cn05 and Cn07. A foreign (other manufacturer's) card generally fails this test and so is disqualified in the Pascal system's list of active slots. Let us say, for example, that the Apple user has installed a foreign printer card in slot 1, and attempts to do a WRITELN to the printer. Pascal formats the request and sends it up the interpreter, which in turn reformats it and sends it to the low-level I/O package, which in Apple Pascal is called the BIOS for Basic I/O Subsystem. At the BIOS level, the slot list is checked to see what kind of card is in slot 1 (the only legal slot for the printer). It is at this point that the output request fails, since slot 1 is marked invalid unless an Apple printer, com, or serial card is in place. It is important to note in what follows that the BIOS driver routine is the only routine in the system that accesses the slot.

The BIOS

The term BIOS not only refers to the set of I/O drivers installed in the file SYSTEM.APPLE in the Pascal system, but also to the written specification for these drivers. This precise specification of the input and output to Pascal was created by the University of California, San Diego (UCSD) to make it easy to interface UCSD Pascal (parent of Apple Pascal) to any number of machines and machine configurations. The type of things specified in the BIOS includes data and control parameters, calling sequence, unit numbers, and error handling requirements.

For example, an Apple-specific version of the BIOS for the printer contains the following information:

- (1) Transfer to the printer device is one character at a time.

- (2) The following special characters must be recognized:
- CR (carriage return) (hex 0D) Print the line and return the carriage to the first column. An automatic line feed should NOT be done.
 - LF (line feed) (hex 0A) Sent only after a CR. Should do a simple line feed (no return) if possible, else a CRLF.
 - FF (form feed) (hex 0C) Advance the paper to top-of-form (if possible) and perform a carriage return.
- (3) There are only two calls to the Printer BIOS, a write and an initialization call. The initialization call should perform carriage return-line feed IF desired, but should not do a form-feed. The printer buffer should be flushed.

Interface Routines	Parameters
Printer write	data byte in register A return completion code in register X (zero if no error)
Printer init	completion code in register X (zero if online, nine if offline)

The completion code is identical to the Pascal IORESULT.

(End of Printer BIOS)

A complete description of the BIOS specification is beyond the scope of this document. The BIOS specification is available at cost from the UCSD Pascal distributor (Softtech Microsystems of San Diego).

Changing the BIOS

At Apple, a BIOS has been written according to the UCSD specification to interface with all Apple peripherals. It is possible to modify the BIOS module within SYSTEM.APPLE to accommodate foreign types of peripheral cards, or to alter the behavior of an existing device such as the screen. This document contains the information necessary to enable one to manipulate the BIOS in the Pascal system in order to add non-Apple devices.

Pascal units for I/O drivers

For many types of additional hardware, such as an external clock, it is better not to alter the BIOS but to construct a Pascal unit containing routines to interface with the hardware. This unit (which may consist of assembly language routines) accesses the Cxxx addresses directly to initialize, command, and retrieve data from the hardware. The Pascal user then simply "uses" the unit and has available the routines he needs, without a system reconfiguration. (Note: The Cxxx address space may be accessed directly from Pascal with a "trix" record:

```

DEVICEBYTES = PACKED ARRAY [0..n] OF CHAR;
TRIX = RECORD CASE BOOLEAN OF
    FALSE: (ADDR: INTEGER);
    TRUE : (CONTROLREC: ^ DEVICEBYTES)

```

```

END;
VAR CARD: TRIx;
One then accesses data by the sequence
CARD.ADDR := (address of hardware);
CARD.CONTROLREC^[0] := CHR(INITMASK)      etc.
The type CHAR is used to ensure that only the desired byte is
accessed (not the whole word).

```

Patching the BIOS.

Some applications such as printers really are required to be built into the system (i.e. so that WRITELN statements access the proper output device). APPLE PASCAL is aware of the following I/O devices: CRT screen, keyboard, printer, graphics output, remote in, remote out, and block devices (disks). Remote input and remote output are general ports for serial or parallel communication. Any device that fits into one of these categories can be handled by APPLE PASCAL, by inserting a customized I/O driver into the BIOS.

The procedure to do so consists of the following steps:

1. Write the I/O driver according to the BIOS specification.
2. Patch the I/O driver into SYSTEM.APPLE at a specified location (see below).
3. Patch the appropriate BIOS vector in SYSTEM.APPLE.

The "specified locations" in step 2 are in two parts. First, one may make use of the space used by the existing driver if there is one. These spaces are listed in Table I below. Generally, it is preferable to use this space in order to avoid wasting it.

If the space shown in Table I is insufficient, a small amount of unused space exists at locations hex DABE - DB7F. This is block 5, byte 190 to block 5, byte 383 (decimal), inclusive.

The BIOS vectors to patch to point to the new routine are given by Table II. Note that the method of routine entry is via a JSR, with the return address on the stack.

The actual patching of SYSTEM.APPLE must be done by a user-supplied patcher; an easy method is to use BLOCKREAD to read in the SYSTEM.APPLE file and the new driver object file, then MOVELEFT to copy the new driver in and BLOCKWRITE to write the new interpreter file. A supplier should provide to the user, as a package, such a program along with his hardware and altered BIOS routine, making it easy for an Apple user to add several modifications to his single Apple Pascal system.

Folding of the Language card memory is taken care of automatically by the interpreter.

When writing BIOS routines, there is one additional consideration: the keyboard input to Apple Pascal is done by polling, not by interrupts; therefore, at convenient locations throughout the BIOS there exist calls to the character-available check routine located at location D681 in the interpreter. When replacing the BIOS routines with customized counterparts, those substitute routines should each contain a call to the check routine.

TABLE I BIOS DRIVER AREAS IN FILE SYSTEM.APPLE

AREA	RAM LOCATION		SYSTEM.APPLE FILE	
	LO-ADDR	HI-ADDR	LO-BLK, BYTE	HI-BLK, BYTE
CONSOLE INIT & READ	D681	D787	3,129	3,391
CONSOLE WRITE	D7D0	D7F6	3,464	3,502
SCREEN DRIVER	D87B	DABD	4,123	5,189
PRINTER INIT	D788	D790	3,392	3,400
PRINTER WRITE	D830	D84D	4,48	4,77
REMOTE INIT	D79C	D7A2	3,412	3,418
REMOTE READ	D84E	D87A	4,78	4,122
REMOTE WRITE	D809	D810	4,9	4,16
COMMON ROUTINES FOR	D791	D79B	3,401	3,411
PRINTER & REMOTE INIT	D7A3	D7CF	3,419	3,463
SERIAL CARD WRITE	D7F7	D808	3,503	4,8
PRINTER CARD WRITE	D811	D81E	4,17	4,30
COM CARD WRITE	D81F	D82F	4,31	4,47
DISK ROUTINES	D000	D569	0,0	2,361
GRAPHIC WRITE	none			

Notes on Table I.

RAM addresses are given in hex. Locations in the file SYSTEM.APPLE are given in decimal as a block & byte offset from the beginning of the file (starting at zero).

Console init routines contain routines used by various other parts of the system, including console write. It is best not to destroy these. Replacement of the Apple keyboard will entail replacement of the console character available check routine mentioned above, with a hook at the original location. Note that the console is highly reconfigurable from the PASCAL level (program SETUP); thus, one should be very thoughtful before putting out the effort to change the console BIOS.

Console write calls the screen driver, unless an external terminal is attached.

Both the printer and remote init routines use both of the common routines listed.

Printer write uses printer card write, serial card write, and com card write. Remote write uses serial card write, com card write, and printer write (sorry about the nonmodularity).

The console and disk drivers are considered replaceable and not extendable. Therefore, the (large) space allotted in Table I is considered sufficient for console and disk needs. Users wishing to interface with the existing console or disk drivers should do so with the cooperation of Apple Computer.

TABLE III BIOS VECTORS

VECTOR	RAM LOCATION	SYSTEM.APPLE FILE BLK, BYTE
CONSOLE READ	FF7B	23,379
CONSOLE WRITE	FF84	23,388
CONSOLE INIT	FF8D	23,397

PRINTER WRITE	FF96	23,406
PRINTER INIT	FF9F	23,415
DISK WRITE	FFA8	23,424
DISK READ	FFB1	23,433
DISK INIT	FFBA	23,442
REMOTE READ	FFC3	23,451
REMOTE WRITE	FFCC	23,460
REMOTE INIT	FFD5	23,469
GRAPHICS WRITE	FFDE	23,478

=====

All vectors are the 16-bit argument of a JSR instruction. The address given is the lo-byte address; the high byte goes in the (given address) + 1.

Zero Page Usage

If zero page storage is needed, the following information is useful. Pascal makes use of variables stored at \$50 thru \$FF. The disk routines use variables at \$36-\$4F. Locations \$00-\$35 are considered to be pure temporaries, i.e. information may be stored in these locations but may be destroyed by other routines. Also, locations \$200-\$399 form a temporary area used by the disk and screen-scrolling routines, and may be used as a temporary buffer space for any other routine.

It should be noted that the drawback to all this patching is that it is dependent on the current system version. It is Apple's intention, however, to provide a standard, generalized method for system reconfiguration before the next system release occurs.

BIOS listing

Attached is a listing of the Apple Pascal BIOS, which provides the exact specification of the inputs and outputs of the drivers.

```

/PAGE - 0
Current memory available! 10654
0000:
0000:
0000: .ABSOLUTE
0000:
2 blocks for procedure code 9907 words left

```

```

0000: .PROC BIOS
Current memory available: 10129
0000:
0000: ;-----
0000: ;
0000: ; APPLE BIOS FOR USCD PASCAL
0000: ;
0000: ; COPYRIGHT 1978 APPLE COMPUTER INC
0000: ; WRITTEN BY BILL ATKINSON
0000: ;
0000: ;-----
0000: ;
0000: ; ZERO PAGE PURE TEMPS
0000: ;
0000: ;-----
0000: 0000 ZEROL .EQU 0
0000: 0001 ZERON .EQU 1
0000: 0002 JUMP1 .EQU 2
0000: 0003 JUMP2 .EQU 3
0000: 0004 BXS1L .EQU 4
0000: 0005 BXS1H .EQU 5
0000: 0006 BXS2L .EQU 6
0000: 0007 BXS2H .EQU 7
0000: 0008 CKPTL .EQU 8
0000: 0009 CKPTRH .EQU 9
0000: 000A CHECKL .EQU 10.
0000: 000B CHECKH .EQU 11.
0000: 000C TT1 .EQU 12.
0000: 000D TT2 .EQU 13.
0000: 000E TT3 .EQU 14.
0000:
0000: ;-----
0000: ;
0000: ; ZERO PAGE PERMANENTS
0000: ;
0000: ;-----
0000: 00F0 FIRST .EQU 0F0 ;START ZERO PAGE USE
0000: 00F0 BAS1L .EQU FIRST ;SCREEN 1 POINTER
0000: 00F1 BAS1H .EQU FIRST+1 ;
0000: 00F2 BAS2L .EQU FIRST+2 ;SCREEN 2 POINTER
0000: 00F3 BAS2H .EQU FIRST+3 ;
0000: 00F4 CH .EQU FIRST+4 ;HORIZ CURSOR 0.79
0000: 00F5 CV .EQU FIRST+5 ;VERT CURSOR 0.23
0000: 00F6 TEMP1 .EQU FIRST+6
0000: 00F7 TEMP2 .EQU FIRST+7
0000: 00F8 SYSDM .EQU FIRST+8 ;2 BYTES POINTER OF SYSDM AREA
0000:
0000: ;-----
0000: ;
0000: ; $BFO0 PAGE PERMANENTS
0000: ;
0000: ;-----
0000: BFOE SCRMODE .EQU 0BFC
0000: BFOF LFFLAG .EQU 0BFC
0000: BF11 NLEFT .EQU 0BF11

```

```

0000: BF12      ESCNT      .EQU 0BF12
0000: BF13      RANDL      .EQU 0BF13
0000: BF14      RANDH      .EQU 0BF14
0000: BF15      CONFLGS    .EQU 0BF15
0000: BF16      BREAK      .EQU 0BF16      ;2 BYTES
0000: BF18      RPTR       .EQU 0BF18      ;1 BYTE  READ POINTER
0000: BF19      WPTR       .EQU 0BF19      ;1 BYTE  WRITE POINTER
0000: BF1A      RETL       .EQU 0BF1A
0000: BF1B      RETH       .EQU 0BF1B
0000:
0000:
0000:
0000:      ; -----
0000:      ;
0000:      ; MISCELLANEOUS PROGRAM EQUATES
0000:      ;
0000:      ; -----
0000: 0200      BUFFER      .EQU 0200      ;TEMP HSHIFT BUFFER (OVERLAPS DISK BUF)
0000: 03B1      CONBUF      .EQU 03B1      ;78 CHAR TYPE-AHEAD BUFFER
0000: 004E      CBUFLEN     .EQU 04E      ;78 DECIMAL
0000: D03C      DWRITE      .EQU 0D03C
0000: D040      DREAD       .EQU 0D040
0000: D004      DINIT       .EQU 0D004
0000: D017      DRESET      .EQU 0D017      ;POWERUP DISK INIT
0000: D152      GOFORT      .EQU 0D152      ;PASCAL STARTUP POINT
0000: 000C      NCTRLS      .EQU 12      ;# CTRL CHARS IN TABLE
0000: BFF8      SLTTPS      .EQU 0BFF8
0000:
0000:
0000:
0000:      ; -----
0000:      ;
0000:      ; DISK ROUTINES ARE AT D000 TO D50B
0000:      ; PUT STARTUP STUFF AFTER THEM
0000:      ;
0000:      ; -----
0000:      .ORG 0D5DC
0000:
0000:      ; -----
0000:      ;
0000:      ; HARD RESET INITIALIZATION
0000:      ;
0000:      ; -----
0000: D5DC: D8      RESET      CLD      ;SET HEX MODE
0000: D5DD:
0000: D5DD:
0000: D5DD:      ;
0000: D5DD:      ; CLEAR ALL MEMORY 0 TO BFFF
0000: D5DD:      ;
0000: D5DD:      ; -----
0000: D5DD: A9 00      START      LDA #0
0000: D5DF: 85 00      STA ZEROH
0000: D5E1: 85 01      STA ZEROH
0000: D5E3: A8      TAX
0000: D5E4: AA      TAX
0000: D5E5: 91 00      ZERLP      STA (ZEROH),Y  ;WRITE A BYTE OF 0
0000: D5E7: C8      INY      ;BUMP POINTER
0000: D5E8: D0FB      BNE ZERLP   ;LOOP TILL NEXT PAGE
0000: D5EA: E6 01      INC ZEROH   ;BUMP MDS POINTER
0000: D5EC: E8      INX

```



```

D618: DD 08D6      CMP CN05BYTES-2,X;MATCH TABLE
D61E: D0##        BNE TRYNEXT      ;NO, TRY NEXT IN LIST
D620: A0 07        LDY #7
D622: B1 08        LDA (CKPTRL),Y ;TEST CN07 BYTE
D624: DD 0FD6      CMP CN07BYTES-2,X;MATCH TABLE?
D627: F0##        BEQ STOR        ;BOTH MATCHED. CARD RECOGNIZED
D629:
D61E# 00
D629: CA          TRYNEXT        DEX          ;BUMP TO NEXT IN LIST
D62A: E0 02        CPY #2          ;TRY ALL TYPES IN LIST
D62C: B0E9        BCS NXTYP        ;IF NOT IN LIST,FALL THRU WITH X=1
D62E:
D627# 00
D62E: A4 09        STOR          LDY CKPTRH
D630: BA          TXA
D631: 99 38BF      STA SLTTYP5-0C0,Y
D634:
D609# 00
D605# 00
D601# 00
D634: A4 09        NOPROM        LDY CKPTRH
D636: 88          DEY          ;BUMP TO NEXT LOWER SLOT
D637: C0 C0        CPY #0C0      ;SLOT 7 DOWNT0 1 DONE?
D639: D0B8        BNE NXTCRD     ;LOOP TILL 7 SLOTS DONE
D63B:
D63B: ;-----
D63B: ;
D63B: ; SET SCREEN MODE ETC
D63B: ;
D63B: ;-----
D63B: AD 51C0      LDA 0C051      ;SET TEXT MODE
D63E: AD 52C0      LDA 0C052      ;SET BOTTOM 4 GRAFIX
D641: AD 54C0      LDA 0C054      ;SELECT PRIMARY PAGE
D644: AD 57C0      LDA 0C057      ;SELECT HTRES GRAFIX
D647: AD 10C0      LDA 0C010      ;CLEAR KEYBOARD STROBE
D64A: 20 ###      JSR FORM        ;ERASE SCREEN
D64D: 20 ###      JSR INVERT      ;PUT CURSOR ON SCREEN
D650: 20 17D0      JSR DRESET      ;DO ONCE ONLY DISK INIT
D653: AD F8BF      LDA SLTTYP5+3 ;WHAT CARD IN SLOT 3
D656: A0 30        LDY #030      ;SLOT 3
D658: 20 ###      JSR CENIT       ;SET BAUD IF COMMUNICATION OR SERIAL THERE
D65B: E0 00        CPY #0         ;WAS AN EXTERNAL CONSOLE THERE?
D65D: D0##        BNE STARTUP     ;NO,USE APPLE SCREEN
D65F: A9 04        LDA #4
D661: 8D 0EBF      STA SCRMODE    ;SET BIT 2 FOR EXTERNAL CONSOLE
D65D# 00
D664: 4C ###      STARTUP        JMP JPASCAL ;FOLD IN INTERP AND START PASCAL
D667:
D667: ;-----
D667: ;
D667: ; SUB TO CHECKSUM ONE PAGE
D667: ;
D667: ;-----
D667:
D5FD# 67D6
D5F6# 67D6

```



```

D667: A9 00      CKPAGE      LDA #0
D669: AA          TAX          ;CLEAR SUM
D66A: AB          TAY          ;CLEAR INDEX
D66B: 18          CKNX        CLC
D66C: 71 03      ADC (CKPTR),Y ;ADD BYTE
D66E: 90 01      BCC NOCRY
D670: E8          INC         ;INC HI BYTE IF CARRY
D671:

D672: 00          NOCRY       INY          ;BUMP INDEX
D673: CB          BNE CKNX     ;SUM 256 BYTES
D674: D0F7        RTS         ;RETURN SUM IN X+A AND Y=0
D675:

D675:      .INCLUDE BIOS:DEVICES.TEXT
D675:      ;-----
D675:      ;
D675:      ; BIOS HANDLERS FOR LOGICAL AND PHYSICAL DEVICES
D675:      ;
D675:      ;-----
D675:
D675:
D675:
D675:
D675:
D675:
D675:
D675:
D675:
D675:      ;-----
D675:      ;
D675:      ; OLD MONITOR WAIT ROUTINE
D675:      ; (LOCATION CHANGED)
D675:      ;
D675:      ;-----
D675:
D675: 38          WAIT          SEC
D676: 48          WAIT2       PHA
D677: E9 01      WAIT3       SBC #1
D679: D0FC        BNE WAIT3   ;1.0204 USECS
D67B: 68          PLA         ;(13+2712+A+512+A)
D67C: E9 01      SBC #1
D67E: D0F6        BNE WAIT2
D680: 60          RTS
D681:
D681:
D681:      ;-----
D681:      ;
D681:      ; CONSOLE CHECK FOR CHAR AVAIL
D681:      ; STATUS AND ALL REG PRESERVED
D681:      ; IF CHAR AVAIL, PUT IN CONBUF AND INC WPTR
D681:      ;
D681:      ; WARNING...THIS ROUTINE ALSO CALLED FROM DISK ROUTINES
D681:      ;
D681:      ;-----
D681:
D681: 09          CONCK          PHP
D682: 48          PHA
D683: 8A          TXA

```

D684: 48		PHA	
D685: 98		TYA	
D686: 48		PHA	
D687: EE 13BF	RNDTMC	INC RANDL	;BUMP 16 BIT RANDOM SEED
D68A: D0**		BNE RNDOK	
D68C: EE 14BF		INC RANDH	
D68A* 00			
D68F: AD FBBF	RNDOK	LDA SLTTPS+3	;WHAT CARD IS IN SLOT 3?
D692: C9 03		CMP #3	;IS IT A COMM CARD?
D694: F0**		BEQ COMCK	;YES, GOT CHECK IT
D696: C9 04		CMP #4	;IS IT A SERIAL CARD?
D698: F0**		BEQ JDONCK	;YES, IT CAN'T BE TESTED
D69A: AD 00C0	TSTKBD	LDA 0C000	;TEST APPLE KEYBOARD
D69D: 10**		BPL JDONCK	;NO CHAR AVAIL
D69F: 8D 10C0		STA 0C010	;CLEAR KEYBD STROBE
D6A2: 29 7F		AND #07F	;MASK OFF TOP BIT
D6A4: C9 0B		CMP #11,	;CTRL-K?
D6A6: D0**		BNE NOTK	
D6A8: A9 5B		LDA #05B	;YES,REPACE WITH LEFT SQR BRACKET
D6A6* 00			
D6AA: C9 01	NOTK	CMP #1	;CTRL-A?
D6AC: D0**		BNE HTTAB	
D6AE: 20 ****		JSR HTAB	;YES,TAB NEXT MULT 40
D6B1: AD 15BF		LDA CONFLGS	
D6B4: 29 FE		AND #0FE	
D6B6: 8D 15BF		STA CONFLGS	;CLEAR AUTO-FOLLOW BIT
D6B9: 4C ****		JMP DONECK	
D6AC* 00			
D6BC: C9 1A	NTTAB	CMP #26,	;CTRL-Z?
D6BE: D0**		BNE NOTFOL	;NO. PUT CHAR IN BUFFER
D6C0: AD 15BF		LDA CONFLGS	
D6C3: 09 01		ORA #1	
D6C5: 8D 15BF		STA CONFLGS	;SET AUTO-FOLLOW BIT
D6C8: D0**		BNE DONECK	;BR ALWAYS
D6CA:			
D694* 00			
D6CA: AD BEC0	COMCK	LDA 0C0BE	;CHAR AVAIL?
D6CD: 4A		LSR A	
D6CE: 90**		BCC DONECK	;NO CHAR AVAIL
D6D0: AD BFC0		LDA 0C0BF	;GET CHAR FROM UART
D6D3: 29 7F		AND #07F	;MASK OFF BIT 7
D6D5:			
D68E* 00			
D6D5: A0 55	NOTFOL	LDY #055	
D6D7: D1 F8		CMP (SYSCON),Y	;STOP CHAR?
D6D9: D0**		BNE NOTSTOP	
D6DB: AD 15BF		LDA CONFLGS	
D6DE: 49 80		EOR #080	
D6E0: 8D 15BF		STA CONFLGS	;YES,TOGGLE STOP BIT(BIT 7)
D6E3:			
D69D* 00			
D698* 00			
D6E3: 4C ****	JDONCK	JMP DONECK	
D6E6:			
D6D9* 00			

```

D6E6: 89          NOTSTOP  DEY
D6E7: D1 F8      CMP (SYSCON),Y
D6E9: D0**       BNE NOTBRK
D6EB: AD 15BF    LDA CONFLGS
D6EE: 29 3F      AND #03F
D6F0: 8D 15BF    STA CONFLGS      ;CLEAR FLUSH&STOP BITS
D6F3: 6C 16BF    JMP @BREAK      ;BREAK OUT
D6F6:
D6E9* 00
D6F6: 88          NOTBRK  DEY
D6F7: D1 F8      CMP (SYSCON),Y      ;FLUSH?
D6F9: D0**       BNE NOTFLUS
D6FB: AD 15BF    LDA CONFLGS
D6FE: 49 40      EOR #040
D703: 8D 15BF    STA CONFLGS      ;TOGGLE FLUSH BIT(BIT 7)
D706: 4C ****    JMP DONECK
D706:
D6F9* 00
D706: AE 19BF    NOTFLUS  LDX WPTR
D709: 20 ****    JSR BUMP
D70C: EC 18BF    CPX RPTR      ;BUFFER FULL?
D70F: D0**       BNE BUFOK
D711: 20 ****    JSR BELL      ;BEEP & IGNORE CHAR
D714: 4C ****    JMP DONECK
D70F* 00
D717: 8E 19BF    BUFOK    STX WPTR
D71A: 9D B103    STA CONBUF,X    ;PUT CHAR IN BUFFER
D71D:
D715* 10D7
D704* 10D7
D6E4* 10D7
D6CE* 00
D6C8* 00
D6BA* 10D7
D71D: 2C 15BF    DONECK  BIT CONFLGS      ;IS STOP FLAG SET?
D720: 10**       BPL CXEXIT
D722: 4C 87D6    JMP RNDINC      ;LOOP IF IN STOP MODE
D720* 00
D725: 63          CXEXIT  PLA
D726: AB          TAY
D727: 68          PLA
D728: AA          TAX
D729: 68          PLA
D72A: 28          PLP
D72B: 60          RTS      ;ELSE RESTORE STAT AND ALL REG AND RETURN
D70A* 2CD7
D72C: E8          BUMP    INX      ;BUMP BUFFER POINTERS WITH WRAP AROUND
D72D: E0 4E      CPX #CBUFLEN
D72F: D0**       BNE DWPRTS
D731: A2 00      LDX #0
D72F* 00
D733: 60          DWPRTS  RTS
D734:
D734:
D734: ;-----

```

```

0734: ;
0734: ; INITIALIZE CONSOLE:
0734: ;
0734: ;-----
0734: 68 CINIT PLA
0735: 85 F6 STA TEMP1 ;SAVE RETURN ADDR
0737: 68 PLA
0738: 85 F7 STA TEMP2
073A: 68 PLA
073B: 85 F8 STA SYSCOM ;SAVE POINTER TO SYSCOM AREA
073D: 68 PLA
073E: 85 F9 STA SYSCOM+1
0740: 68 PLA
0741: 80 16BF STA BREAK ;SAVE BREAK ADDRESS
0744: 68 PLA
0745: 80 17BF STA BREAK+1
0749: A5 F7 LDA TEMP2
074A: 48 PHA ;RESTORE RETURN ADDRESS
074B: A5 F6 LDA TEMP1
074D: 48 PHA
074E: AD 18BF LDA RPTR ;FLUSH TYPE-AHEAD BUFFER
0751: 80 19BF STA WPTR
0754: AD 15BF LDA CONFLGS
0757: 29 3E AND #03E
0759: 80 15BF STA CONFLGS ;CLEAR STOP,FLUSH,AUTO-FOLLOW BIT
075C: 20 *** JSR TAB3 ;NO,HORIZ SHIFT FULL LEFT
075F: A2 00 LDX #0 ;CLEAR IORESULT
0761: 60 RTS ;AND RETURN
0762: ;-----
0762: ;
0762: ; READ FROM CONSOLE:
0762: ; KEYBOARD,COM OR SERIAL CARD IN SLOT 3
0762: ;
0762: ;-----
0762: 20 *** CREAD JSR ADJUST ;HORIZ SCROLL IF NECESSARY
0765: A0 30 LDY #030 ;SLOT 3
0767: AD FBBF LDA SLTTPS+3 ;WHAT TYPE OF CARD?
076A: C9 04 CMP #4 ;IS IT A SERIAL CARD?
076C: D0** BNE CREAD2 ;NO,CONTINUE
076E: 4C *** JMP RSER ;YES,READ IT
076C: 00
0771: 20 B1D6 CREAD2 JSR CONCK ;TEST CHAR
0774: AE 18BF LDX RPTR
0777: EC 19BF CPX WPTR
077A: F0E6 BEQ CREAD ;LOOP TILL SOMETHING IN BUFFER
077C: 20 2CD7 JSR BUMP
077F: 8E 18BF STX RPTR ;BUMP READ POINTER
0782: BD B1D3 LDA CONBUF,X ;GET CHAR FROM BUFFER
0785: A2 00 LDX #0 ;CLEAR IORESULT
0787: 60 RTS ;AND RETURN TO PASCAL
0789: ;-----
0789: ;
0789: ; INITIALIZE PRINTER:
0789: ; PRINTER IS ALWAYS IN SLOT 1
0789: ; IT MAY BE A PRINTER,COM. OR SERIAL CARD

```

```

D788:
D788:
D788: A0 10      PINIT      LDY #010      ;SLOT 1
D78A: AD F9BF    LDA SLTTPS+1    ;WHAT CARD IN SLOT 1?
D78D: C9 05      CMP #5          ;PRINTER CARD?
D78F: F0**      BEQ CLR101      ;YES, NO INIT NEEDED
D659* 91D7
D791: C9 04      CENT          CMP #4      ;SERIAL CARD?
D793: F0**      BEQ ISER        ;YES, INIT SER CARD
D795: C9 03      CMP #3          ;COM CARD?
D797: F0**      BEQ ICOM        ;YES, INIT COM CARD
D799: A2 09      LDX #9          ;NONE OF ABOVE, OFF LINE
D79B: 60        RTS
D79C:
D79C:
D79C:           ; INITIALIZE REMOTE:
D79C:           ; REMOTE IS ALWAYS IN SLOT 2
D79C:           ; IT MAY BE A COM OR SERIAL CARD
D79C:
D79C:
D79C:
D79C: AD FAFB    RINIT      LDA SLTTPS+2    ;WHAT CARD IN SLOT 2?
D79F: A0 20      LDY #020
D7A1: D0EE      BNE CENT        ;BR ALWAYS TAKEN
D7A3:
D7A3:
D7A3:           ; INIT COM CARD, Y=0ND
D7A3:
D7A3:
D7A3:
D797* 00
D7A3: A9 03      ICOM          LDA #3        ;MASTER INIT
D7A5: 99 8EC0    STA 0C0BE,Y      ;TO STATUS
D7A8: A9 15      LDA #21.
D7AA: 99 8EC0    STA 0C0BE,Y      ;SET BAUD ETC
D7B* 00
D7AD: A2 00      CLR101      LDX #0        ;CLEAR IORESULT
D7AF: 60        RTS            ;AND RETURN
D7B0:
D7B0:
D7B0:           ; INIT SERIAL CARD, Y=0ND
D7B0:
D7B0:
D7B0:
D793* 00
D7B0: 20 ***
D7B3: 20 00C8
D7B6: A2 00      CLRT03      LDX #0        ;CLEAR IORESULT
D7B8: 60        RTS            ;AND RETURN
D7B9:
D7B9:
D7B9:           ; ASSORTED SERIAL CARD SET-UP
D7B9:
D7B9:
D7B9:
D7B1* B9D7
D7B9: 8C F806    SER1         STY 06F8      ;STORE NO
D7BC: 98        TYA
D7BD: 4A        LSR A

```

```

07BE: 4A          LSR A
07BF: 4A          LSR A
07C0: 4A          LSR A
07C1: 09 C0       ORA #0C0
07C3: AA          TAX          ;MAKE OCN IN X
07C4: A9 00       LDA #0
07C6: 85 F6       STA TEMP1
07C8: 86 F7       STX TEMP2      ;SET UP INDIRECT ADDRESS
07CA: AD FCF      LDA 0CFF      ;TURN OFF ALL CB ROMS
07CD: B1 F6       LDA (TEMP1),Y  ;SELECT CB BANK
07CF: 60          RTS

07D0:             ;-----
07D0:             ;
07D0:             ; WRITE TO CONSOLE
07D0:             ; VIDEO SCREEN, COM OR SER CARD IN SLOT 3
07D0:             ;
07D0:             ;-----
07D0: 20 81D6     CURITE      JSR CONCK      ;CONSOLE CHAR AVAIL?
07D3: 2C 15BF     BIT CONFLGS  ;IS FLUSH FLAG SET?
07D5: 7C**        BVS CLRIO     ;YES, DISCARD CHAR & RETURN
07D8: AA          TAX          ;SAVE CHAR IN X
07D9: A0 30       LDY #30      ;SLOT 3, C10
07DB: AD F8BF     LDA SLTTP3+3  ;WHAT KIND OF CARD?
07DE: C9 03       CMP #3       ;COM CARD?
07E0: F0**        BEQ WCOM      ;YES, WRITE TO COM CARD SLOT 3
07E2: C9 04       CMP #4       ;SERIAL CARD?
07E4: F0**        BEQ USER     ;YES, WRITE TO SER CARD SLOT 3
07E6: 8A          TXA          ;ELSE RESTORE CHAR AND SEND TO SCREEN
07E7: 85 F6       STA TEMP1     ;SAVE CHAR FOR LATER
07E9: 20 ***      JSR INVERT    ;REMOVE CURSOR
07EC: A4 F4       LDY CH
07EE: 20 ***      JSR VOUT2     ;DO THE BUSINESS
07F1: 20 ***      JSR INVERT    ;RESTORE THE CURSOR
07D6* 03
07F4: A2 00       CLRIO        LDX #0      ;CLR IO RESULT
07F6: 60          RTS          ;RETURN FROM VIOOUT

07F7:             ;-----
07F7:             ;
07F7:             ; WRITE TO SERIAL CARD, Y=0ND, CHAR IN X
07F7:             ;
07F7:             ;-----
07E4* 00
07F7: 20 81D6     USER        JSR CONCK      ;CONSOLE CHAR?
07FA: 8A          TXA
07FB: 48          PHA          ;SAVE CHAR ON STACK
07FC: 20 B9D7     JSR SER1      ;ASSERTED GARBAGE
07FF: 68          PLA
0800: 9D B805     STA 0583,X     ;SET UP DATA BYTE
0803: 20 AAC9     JSR 029AA     ;SEND IT (SHOUT)
0806: A2 03       LDX #3
0808: 60          RTS

0809:             ;-----
0809:             ;
0809:             ; WRITE TO REMOTE: CHAR IN A
0809:             ;

```

```

D809:
D809: AA
D80A: AD FAFB
D80D: A0 20
D80F: D9**
D811:
D811:
D811:
D811:
D811:
D811: 20 8106
D814: AD C1C1
D817: 30F8
D819: 8E 90C0
D81C: A2 00
D81E: 60
D81F:
D81F:
D81F:
D81F:
D81F:
D81F:
D7E0* 00
D81F: 20 8106
D822: B9 8E00
D825: 29 02
D827: F0F6
D829: 8A
D82A: 99 8FC0
D82D: A2 00
D82F: 60
D830:
D830:
D830:
D830:
D830:
D830: AA
D831: AD 0FBF
D834: 10**
D836: E0 0A
D838: F0BA
D83A:
D83A* 00
D83A: A0 10
D83C: AD F9BF
D83F: C9 05
D841: F0CE
D80F* 00
D843: C9 04
D845: F0B0
D847: C9 03
D849: F0D4
D84B:
D84B:
D84B: A2 09

```

```

;-----
RWRITE TAX ;SAVE CHAR
LDA SLTTYP$+2 ;WHAT CARD IN SLOT 2?
LDY #020
BNE CENW2 ;BR ALWAYS TAKEN
;-----
;
; WRITE TO PRINTER CARD SLOT 1, CHAR IN X
;
;-----
WPRN JSR CONCK ;CONSOLE CHAR AVAIL?
LDA 0C1C1 ;TEST PRINTER READY
BMT WPRN ;LOOP TILL READY
STX 0C090 ;SEND CHAR
CLRT02 LDX #0
RTS
;-----
;
; WRITE TO COM CARD. Y=0ND CHAR IN X
;
;-----
WCOM JSR CONCK ;CONSOLE CHAR?
LDA 0C03E,Y ;TEST UART STATUS (ACIA STATUS REGISTER)
AND #2 ;READ?
BEQ WCOM ;NO, WAIT TILL READY
TXA
STA 0C03E,Y ;SEND CHAR
LDX #0
RTS
;-----
;
; WRITE TO PRINTER: CHAR IN A
;
;-----
PWRITE TAX ;SAVE CHAR IN X
LDA LFFLAG ;TEST LINE-FEED FLAG
BPL LFPASS ;PASS IF BIT7=0
CPX #10. ;IS IT A LINE-FEED?
BEQ CLRT0 ;YES, IGNORE
LFPASS LDY #010 ;SLOT 1
LDA SLTTYP$+1 ;WHAT KIND OF CARD?
CENW CMP #5 ;PRINTER CARD?
BEQ WPRN ;YES WRITE TO PRINTER CARD
CENW2 CMP #4 ;SERIAL CARD?
BEQ USER ;YES WRITE TO SER CARD
CMP #3 ;COM CARD?
BEQ WCOM ;YES WRITE TO COM CARD
OFFLINE LDX #9

```



```

B87B: ; (APPLE SCREEN EMULATES A DATAMEDIA TERMINAL)
B87B: ;
B87B: ;-----
B87B: 1B CTRLCH .BYTE 27. ;ESCAPE
B87C: 1E .BYTE 30. ;GOTOXY
B87D: 0B .BYTE 13. ;CR
B87E: 0A .BYTE 10. ;LF
B87F: 07 .BYTE 7. ;BELL
B880: 1F .BYTE 31. ;REVLf
B881: 1C .BYTE 28. ;NDfS
B882: 08 .BYTE 8. ;NDfS
B883: 0C .BYTE 12. ;fORM
B884: 19 .BYTE 25. ;fOME
B885: 08 .BYTE 11. ;CLEOS
B886: 1D .BYTE 29. ;CLEOL
B887: ;-----
B887: ;
B887: ; JUMP TABLE FOR CONTROL CHARS
B887: ;
B887: ;-----
B887: 0000 CTRLJMP .WORD ESCAPE
B889: 0000 .WORD GOTOXY
B88B: 0000 .WORD CR
B88D: 0000 .WORD LF
B88F: 0000 .WORD BELL
B891: 0000 .WORD REVLf
B893: 0000 .WORD ADVANCE
B895: 0000 .WORD NDfS
B897: 0000 .WORD fORM
B899: 0000 .WORD fOME
B89B: 0000 .WORD CLEOS
B89D: 0000 .WORD CLEOL
B89F: ;-----
B89F: ;
B89F: ; START OF SCREEN HANDLING ROUTINES
B89F: ;
B89F: ;-----
B7EF: 9FD8
B89F: AD 12BF VOUT2 LDA ESCNT ;STILL IN ESC SEQ?
B8A2: F000 BEQ NOTSEQ ;NO, SKIP
B8A4: C9 02 CMP #2 ;ARE WE IN XY MODE?
B8A6: 9000 BCC SECRET ;NO, THERE ARE NO OTHER ESC SEQUENCES
B8A8: F000 BEQ SETY ;IF ESCNT=2 THEN SET CV
B8AA: A5 F6 LDA TEMP1 ;ELSE GET BACK CHAR
B8AC: 38 SEC
B8AD: E9 20 SBC #32. ;SUB 32
B8AF: 3000 BHI DOPsx ;BRANCH IF NEG
B8B1: C9 50 CMP #80.
B8B3: 9000 BCC XOK
B8BF: 00
B8B5: A9 00 DOPsx LDA #0 ;OUT OF RANGE, MAKE =0
B8B7: 00
B8B9: 85 F7 XOK STA TEMP2 ;SAVE TILL GET Y TO3
B8BB: 4C 0000 JMP SECRET ;DEC ESCNT AND RETURN
B8BD: 00

```

```

D88C: A5 F6      SETY      LDA TEMP1      ;GET BACK CHAR
D88E: 38          SEC
D88F: E9 20      SBC #32,
D8C1: 30xx      BNE D8PSY
D8C3: C9 18      CMP #24,
D8C5: 90xx      BCC Y0A
D8C8: 00
D8C7: A9 00      D8PSY      LDA #0
D8C5: 00
D8C9: 85 F5      Y0A      STA CV      ;SET NEW CV
D8CB: 20 xxxx      JSR BASEAL      ;CALC NEW POINTERS
D8CE: A5 F7      LDA TEMP2      ;SET X COORD FROM LAST
D8D0: 85 F4      STA CH      ;TIME THRU AND SET HORIZ
D8D2: A9 00      LDA #0
D8D4: 8D 12BF     STA ESCNT
D8D7: 60          RTS      ;AND RETURN
D8DA: D8D8
D8A4: 00
D8D8: CE 12BF     SECRET      DEC ESCNT
D8DB: 60          RTS
D8A2: 00
D8DC: A5 F6      NOTSEQ      LDA TEMP1      ;GET CHAR BACK
D8DE: 29 7F      AND #07F      ;MASK OFF HI BIT
D8E0: C9 20      CMP #32,      ;CONTROL CHAR?
D8E2: 90xx      BCC CNTRL      ;YES, PROCESS IT
D8E4: C9 60      CMP #96,      ;FLOWER CASE?
D8E6: 90xx      BCC UPPER
D8E8: E9 20      SBC #32,      ;CONVERT TO UPPER
D8EA: 00
D8EA: 29 3F      UPPER      AND #03F      ;TURN OFF FLASH
D8EC: 09 80      ORA #080      ;MAKE NOT INVERTED
D8EE: 4C xxxx      JMP STDAV      ;PRINT AND RETURN
D8E2: 00
D8F1: A2 08      CNTRL      LDX #CNTRL5-1      ;POINT TO END OF CTRL TABLE
D8F3: DD 78D8     CKCTRL      CMP CTRLCH,X      ;COMPARE CHAR AGAINST TABLE
D8F6: F0xx      BEQ DOCTRL      ;PROCESS IF MATCH
D8F8: CA          DEX
D8F9: 10F8      BPL CKCTRL      ;CHECK ALL CHAR IN TABLE
D8FB: 60          RTS      ;IGNORE IF NOT MATCHED
D8FA: 00
D8FC: 8A          DOCTRL      TXA
D8FD: 0A          ASL A      ;DOUBLE INDEX FOR WORD OFFSET
D8FE: AA          TAX
D8FF: BD 89D8     LDA CTRLJMP+1,X      ;GET HI BYTE Jmp ADDR
D902: 85 03      STA JMP2
D904: BD 87D8     LDA CTRLJMP,X      ;AND LO BYTE
D907: 85 02      STA JMP1
D909: 6C 0200     JMP @JMP1      ;DO TABLE JUMP
D90C: ;-----
D90C: ;
D90C: ; ROUTINES TO PROCESS CONTROL CHARS
D90C: ;
D90C: ;-----
D887: 0CD9

```

B90C: A9 01	ESCAPE	LDA #1	
B90E: 8D 12BF		STA ESCNT	;SET ESCAPE FLAG
B911: 60		RTS	;AND RETURN
D889: 12D9			
D912: A9 03	GOTOX1	LDA #3	
D914: 8D 12BF		STA ESCNT	;SET ESC COUNT=3 FOR X1
B917: 60		RTS	
D88B: 18D9			
D918: A0 00	CR	LDY #0	
B91A: 84 F4		STY CH	;RESET HORIZ CURSOR
B91C: 60		RTS	
D88D: 1DD9			
D91D: A5 F5	LF	LDA CV	;GET VERT CURSOR
D91F: C9 17		CMP #23,	;ON BOTTOM LINE?
D921: F0xx		BEG SCROLL	;YES, SCROLL UP
B923: E6 F5	LF2	INC CV	;NO, BUMP VERT CURSOR
B925: 4C ###		JMP BASCAL	;CALC POINTERS AND RETURN
B928: 00			
D92B: A5 F4	SCROLL	LDA CH	
B92D: 48		PHA	;SAVE HORIZ CURSOR
D92E: A2 00		LDX #0	
D92F: 86 F5		STX CV	;INIT LINE COUNT
B92F: 20 ###		JSR BASCAL	
B932: 20 ###	COPY1	JSR COPY	;COPY POINTERS
B935: E8		INX	
B936: 86 F5		STX CV	;BUMP LINE COUNT
D938: 20 ###		JSR BASCAL	
B93B: A0 27		LDY #39,	
D93D: B1 F0	COPY2	LDA (BAS1L),Y	
D93F: 91 04		STA (BXS1L),Y	;COPY BYTE OF PG1
B941: B1 F2		LDA (BAS2L),Y	
D943: 91 06		STA (BXS2L),Y	;COPY BYTE OF PG2
D945: 88		DEY	
B946: 10F5		BPL COPY2	;LOOP 40 CHARACTERS
B948: E0 17	SAPCOUT	CPX #23,	;LINE 23 COPIED?
B94A: D0E6		BNE COPY1	;NO, LOOP UNTIL PAGE
D94C: A9 00		LDA #0	
D94E: 85 F4		STA CH	;CLEAR HORIZ CURSOR
D950: 20 ###		JSR CLEOL	;CLEAR BOTTOM LINE
D953: 68		PLA	
D954: 85 F4		STA CH	;RESTORE HORIZ CURSOR
B956: 60		RTS	;AND RETURN
D88F: 57D9			
D712: 57D9			
D957: A9 40	BELL	LDA #040	;DELAY .01 SEC
B959: 20 7506		JSR WAIT	
D95C: A0 C0		LDY #0C0	
D95E: A9 0C	BELL2	LDA #09C	;TOGGLE SPEAKER AT
D960: 20 7506		JSR WAIT	;1 KHZ FOR .1 SEC
D963: AD 30C0		LDA 0C030	;TOGGLE SPEAKER
D966: 88		DEY	
B967: D0F5		BNE BELL2	
D969: 60	JRET	RTS	;AND RETURN
B891: 6AD9			
D96A: A5 F5	REVL	LDA CV	;IS CURSOR ALREADY AT TOP?


```

D9R3:      ; FALL THROUGH TO ADVANCE
D9R3:      ; X R:G UNCHANGED. Y=CH ON EXIT
D9R3:      ;
D9R3:      ; -----
D9AC: R109
D8EF: R109
D9R3: 20 ***
D9R6:
D9R6:      ; -----
D9R6:      ;
D9R6:      ; ADVANCE CURSOR TO RIGHT.
D9R6:      ; IF CH ALREADY = 79 THEN DON'T ADVANCE,
D9R6:      ; AND RETURN WITH Y=79 AND CARRY SET
D9R6:      ; ELSE RETURN WITH Y=CH, CARRY CLEAR
D9R6:      ; X & A UNCHANGED
D9R6:      ;
D9R6:      ; -----
D9R6:
D893: R609
D9R6: A4 F4
D9R8: C0 4F
D9R4: B0**
D9BC: C8
D9BD: 84 F4
D9BF:
D9BA: 00
D9BF: 60
D9C0:
D9C0:      ; -----
D9C0:      ;
D9C0:      ; PUT CHAR ON SCREEN, EITHER VISABLE
D9C0:      ; OR INVISABLE PORTION.
D9C0:      ;
D9C0:      ; SPECIAL CASE : IF A=0 THEN JUST INVERT
D9C0:      ; AT THAT LOCATION INSTEAD OF STORING.
D9C0:      ;
D9C0:      ; -----
D9C0:
D9E4: C009
D9C0: 48
D9C1: A5 F4
D9C3: 38
D9C4: E0 11BF
D9C7: 30**
D9C9: C9 28
D9CB: B0**
D9CD: A8
D9CE: 68
D9CF: D0**
D9D1: B1 F0
D9D3: 49 80
D9D5:
D9CF: 00
D9D5: 91 F0
D9D7: 60
D9D7:      ; -----
D9D7:      ;
D9D7:      ; ADVANCE
D9D7:      ; LDY CH
D9D7:      ; CPY #79.
D9D7:      ; BCS ADVRTS
D9D7:      ; INY
D9D7:      ; STY CH
D9D7:
D9D7:      ; -----
D9D7:
D9D7:      ; ADVRTS
D9D7:      ; RTS
D9D7:
D9D7:      ; -----
D9D7:      ;
D9D7:      ; PUT CHAR ON SCREEN, EITHER VISABLE
D9D7:      ; OR INVISABLE PORTION.
D9D7:      ;
D9D7:      ; SPECIAL CASE : IF A=0 THEN JUST INVERT
D9D7:      ; AT THAT LOCATION INSTEAD OF STORING.
D9D7:      ;
D9D7:      ; -----
D9D7:
D9D7:      ; STORE
D9D7:      ; PHA
D9D7:      ; ;SAVE CHAR FOR LATER
D9D7:      ; LDA CH
D9D7:      ; SEC
D9D7:      ; SBC #LEFT ;WHERE IS THE HORIZ CURSOR?
D9D7:      ; BNE OFFLT ; IT'S OFF TO THE LEFT OF VISABLE SCREEN
D9D7:      ; CMP #40.
D9D7:      ; BCS OFFRT ;IT'S OFF TO THE RIGHT OF VISABLE SCREEN
D9D7:      ; TAY
D9D7:      ; ;ELSE IT'S ON VISABLE SCREEN
D9D7:      ; PLA
D9D7:      ; ;GET CHAR BACK
D9D7:      ; BNE STORE1 ;STORE IT UNLESS 0
D9D7:      ; LDA (BASIL),Y ;IF 0 READ SCREEN
D9D7:      ; EOR #030
D9D7:      ; ;INVERT CHARACTER
D9D7:
D9D7:      ; -----
D9D7:
D9D7:      ; STORE1
D9D7:      ; STA (BASIL),Y ;STORE VISABLE CHAR
D9D7:      ; RTS
D9D7:
D9D7:      ; -----

```

```

D9D8:
D9C7: 00
D9D8: 18          OFFLFT    CLC
D9D9: 69 28        ADC #40.
D9D8: 4C ****      JMP STOR2
D9DE:
D9CD: 00
D9DE: 38          OFFRT     SEC
D9DF: E9 28        SBC #40.
D9E1:
D9DC: E1D9
D9E1: A8          STOR2     TAY
D9E2: 68          PLA       ;GET BACK CHAR
D9E3: D0**        BNE STOR2 ;STORE IF NOT 0
D9E5: B1 F2        LDA (BAS2L),Y ;IF ZERO, READ SCREEN
D9E7: 49 80        FOR #080   ;AND INVERT IT
D9E9:
D9E3: 00
D9E9: 91 F2        STOR2     STA (BAS2L),Y ;STORE HIDDEN CHAR
D9EB: 60          RTS
D9EC:
D9EC:
D9EC:
D9EC: ; INVERT CHAR AT CURSOR POSITION
D9EC:
D9EC:
D9EC:
D9EC:
D9EC:
D7F2: ECD9
D7EA: ECD9
D9EC: A9 00        INVERT    LDA #0       ;FLAG TO TELL STORE TO INVERT
D9EE: 4C C0D9      JMP STOR2   ;INVERT AT CH AND RETURN
D9F1:
D9F1:
D9F1: ; CLACULATE BASE ADDRESS POINTERS
D9F1: ; FOR PAGE 1 AND PAGE 2
D9F1: ; ENTER WITH CV IN RANGE 0..23
D9F1: ; EXIT WITH BAS1L,H, AND BAS2L,H SET UP
D9F1: ; X AND Y REGS UNCHANGED
D9F1:
D9F1:
D9F1:
D9A5: F1D9
D987: F1D9
D971: F1D9
D939: F1D9
D930: F1D9
D926: F1D9
D9CC: F1D9
D9F1: A5 F5        BASCAL    LDA CV
D9F3: 4A          LSR A       ;SET CARRY FOR 6 LINES DOWN
D9F4: 29 03        AND #3
D9F6: 09 04        ORA #4
D9F8: 85 F1        STA BAS1H
D9FA: A5 F5        LDA CV
D9FC: 29 18        AND #018

```

```

B9FE: 90xx          BCC BSCL2
BA00: 69 7F          ADC #07F
DA02:
D9FE: 00
DA02: 85 F0          BSCL2   STA BASIL
BA04: 0A             ASL A
BA05: 0A             ASL A
DA06: 05 F0          ORA BASIL
BA08: 85 F0          STA BASIL
BA0A: 85 F2          STA BAS2L
DA0C: A5 F1          LDA BAS1H
BA0E: 18             CLC
BA0F: 69 04          ADC #4
DA11: 85 F3          STA BAS2H   ;POINTER TO PAGE 2
DA13: 60             RTS
DA14:
DA14: ;-----
DA14: ;
DA14: ; COPY BASIL,H INTO BXSIL,H
DA14: ; AND BAS2L,H INTO BXS2L,H
DA14: ; TO MAKE AN EXTRA PAIR OF POINTERS
DA14: ; FOR SCROLLING, ETC.
DA14: ;-----
DA14:
D933: 14DA
DA14: A5 F1          COPY   LDA BAS1H   ;COPY POINTERS
DA16: 85 05          STA BXS1H
DA18: A5 F3          LDA BAS2H
BA1A: 85 07          STA BXS2H
BA1C: A5 F0          LDA BASIL
DA1E: 85 04          STA BXS1L
DA20: 85 06          STA BXS2L   ;NOTE BAS2L ALWAYS SAME AS BASIL
DA22:
DA22: 60             RTS7    RTS
DA23:
DA23: ;-----
DA23: ;
DA23: ; HORIZONTAL SCREEN SHIFT ROUTINE
DA23: ;
DA23: ;-----
DA23:
DA23: AA             HSHIFT  TAX           ;SAVE HORIZ DELTA
DA24: F0FC          BEQ RTS7   ;RETURN IF DELTA=0
DA26: A5 F5          LDA CV
DA28: 48             PHA       ;SAVE CV ON STACK
DA29: 8A             TXA
DA2A: 48             PHA       ;SAVE OFFSET ON STACK
DA2B: 38             SEC
DA2C: E9 28          SBC #40.
DA2E: 10xx          BPL OKDEL.T
BA30: 18             CLC
DA31: 69 50          ADC #80.   ;CALC (DELTA+40) MOD 80
DA33:
DA2E: 00

```

BA33: 48	OKDEL T	PHA	ISAVE ON STACK
BA34: A9 17		LDA #23,	
BA36: 85 F5		STA CV	INIT ROW COUNTER
BA38: 20 F109	LOP1	JSR BASCAL	CALC POINTERS
BA39: A0 27		LDY #39,	
BA39: B1 F0	LOP2	LDA (BAS1L),Y	GET VISABLE CHAR
BA3F: 99 0002		STA BUFFER,Y	COPY INTO LEFT HALF OF BUFFER
BA42: B1 F2		LDA (BAS2L),Y	GET HIDDEN CHAR
BA44: 99 2802		STA BUFFER+40,,Y	COPY INTO RIGHT HALF OF BUFFER
BA47: 88		DEY	
BA48: 10F3		BPL LOP2	REPEAT FOR WHOLE ROW
BA4A: 68		PLA	
BA4B: 48		PHA	
BA4C: AA		TAX	GET (DELTA+40) MOD 80 IN X
BA4B: A0 27		LDY #39,	
BA4F: CA	VISLP	DEX	
BA50: 1088		BPL NOWRP1	
BA52: A2 4F		LDX #79,	BUFFER INDEX WRAP-AROUND
BA54:			
BA508 00			
BA54: BD 0002	NOWRP1	LDA BUFFER,X	GET FROM BUFFER
BA57: B1 F0		STA (BAS1L),X	WRITE VISABLE CHAR
BA59: 88		DEY	
BA5A: 10F3		BPL VISLP	LOOP 40 VISABLE CHARS
BA5C: A0 27		LDY #39,	
BA5E: CA	HTDNL P	DEX	
BA5F: 1088		BPL NOWRP2	
BA61: A2 4F		LDX #79,	BUFFER INDEX WRAP-AROUND
BA63:			
BA5F8 00			
BA63: BD 0002	NOWRP2	LDA BUFFER,X	
BA66: 91 F2		STA (BAS2L),Y	WRITE A HIDDEN CHAR
BA68: 88		DEY	
BA69: 10F3		BPL HTDNL P	LOOP 40 HIDDEN CHARS
BA6B: C6 F5		DEC CV	
BA6B: 10C9		BPL LOP1	REPEAT ALL ROWS IN PAGE
BA6F: 68		PLA	DISCARD DELTA+40 MOD 80
BA70: 68		PLA	GET PLAIN DELTA
BA71: 18		CLC	
BA72: 6D 118F		ADC #LEFT	
BA75: 8D 118F		STA #LEFT	UPDATE COUNT OF CHARS TO LEFT
BA78: 68		PLA	
BA79: 85 F5		STA CV	RESTORE CV
BA7B: 4C F109		JMP BASCAL	AND POINTERS AND RETURN
BA7E:			
BA7E:			
BA7E:			
BA7E:			
BA7E:			
BA7E:			
BA7E:			
BA7E:			
BA7E:			
BA7E:			
BA7E: AD 118F	HTAB	LDA #LEFT	
BA81: C9 14		CMP #20,	
BA83: B088		BCS TAB3	
BA85: A9 28	TAB2	LDA #40,	


```

DA87: DOXX                BNE DOIT      ;ALWAYS TAKEN
DA83: 00
DA89: A9 00              TAB3      LDA #0
DA87: 00
DA88: 38                DOIT      SEC
DA8C: ED 11BF           SBC HLFF      ;CALC DELTA
DA8F: 4C 230A           JMP HSHIFT    ;SHIFT AND RETURN
DA92:
DA92: ;-----
DA92: ;
DA92: ; ADJUST ROUTINE DOES HORIZ
DA92: ; SCROLL TO KEEP CURSOR VIABLE
DA92: ;
DA92: ;-----
DA92:
DA92:
DA92: ADJUST      LDA COMFLG5
DA95: 4A          LSR A              ;IS AUTO-FOLLOW (BIT 1) TRUE?
DA96: 90XX        BCC ADJRTS        ;NO, IGNORE
DA98: A5 F4          LDA CH
DA9A: C9 14          CMP #20.        ;CURSOR LESS THAN 20?
DA9C: B0XX          BCS ADJOK        ;NO, CONTINUE
DA9E: 4C 89DA          JMP TAB3      ;YES, TAB FULL LEFT AND RETURN
DAA1:
DA9C: 00
DAA1: 38          ADJOK      SEC
DAA2: ED 11BF      SBC HLFF
DAA5: 30XX          BNT ADJLFT
DAA7: C9 25          CMP #37.
DAA9: B0XX          BCS ADJRT
DAA3:
DA96: 00
DAA8: 60          ADJRTS     RTS      ;DON'T ADJUST
DAA6:
DAA9: 00
DAA6: A4 F4          ADJRT      LDY CH
DAAE: C0 4D          CPY #77.
DAB0: B0XX          BCS FARRT
DAB2: 38          SEC
DAB3: E9 24          SBC #36.
DAB5: 4C 230A          JMP HSHIFT
DAB8:
DAB8: 00
DAB8: 4C 85DA          FARRT      JMP TAB2
DAB8:
DAA5: 00
DAB8: 4C 230A          ADJLFT     JMP HSHIFT
DABE:
DABE:
DABE: ;INCLUDE BTOS:TOP.TEXT
DABE:
DABE: 00 00 00 00 00 00 00      .ORG      OFF00
FF00:
FF00: ;-----

```

```

FF00:      ;
FF00:      ; TOP PORTION OF BIOS FOLDS IN AND OUT BIOS 0000 BANK
FF00:      ;
FF00:      ;-----
FF00:
FF00:      ;-----
FF00:      ;
FF00:      ; MAIN BIOS JUMP TABLE CALLED FROM INTERPRETER
FF00:      ;
FF00:      ;-----
FF00:
FF00:      ;
FF00: 4C 0000      JMP OCREAD      ;CONSOLE READ
FF03: 4C 0000      JMP OCWRITE     ;CONSOLE WRITE
FF06: 4C 0000      JMP OCINIT      ;CONSOLE INIT
FF09: 4C 0000      JMP OWRITE      ;PRINTER WRITE
FF0C: 4C 0000      JMP OPINT       ;PRINTER INTT
FF0F: 4C 0000      JMP ODWRITE     ;DISK WRITE
FF12: 4C 0000      JMP ODREAD      ;DISK READ
FF15: 4C 0000      JMP ODINIT      ;DISK INIT
FF18: 4C 0000      JMP OREAD       ;REMOTE READ
FF1B: 4C 0000      JMP OWRITE      ;REMOTE WRITE
FF1E: 4C 0000      JMP ORINIT      ;REMOTE INIT
FF21: 4C 0000      JMP OWRITE      ;GRAPHIC WRITE
FF24:
FF24:      ;-----
FF24:      ;
FF24:      ; STRIP LOCAL RETURN ADDRESS
FF24:      ; STRIP PASCAL ADDRESS AND SAVE IN RETL, RETH
FF24:      ; THEN RESTORE LOCAL RET ADDR AND RETURN
FF24:      ; MEANWHILE UNFOLDS BIOS INTO DXXX
FF24:      ;
FF24:      ;-----
FF24:
FF24:
FF24: 85 0C      SAVERET  STA TT1      ;SAVE A REG
FF26: AD 83C0    LDA OCOR3     ;UNFOLDS BIOS INTO DXXX
FF29: 68        PLA
FF2A: 85 0D      STA TT2      ;LOCAL RET ADDRESS
FF2C: 68        PLA
FF2D: 85 0E      STA TT3
FF2F: 68        PLA
FF30: 8D 1ABF    STA RETL     ;PASCAL RETURN ADDRESS
FF33: 68        PLA
FF34: 8D 1BBF    STA RETH
FF37: A5 0E      LDA TT3
FF39: 4B        PHA           ;RESTORE LOCAL RETURN ADDRESS
FF3A: A5 0D      LDA TT2
FF3C: 4B        PHA
FF3D: A5 0C      SKIPSAV  LDA TT1      ;RESTORE A REG
FF3F: 60        RTS          ;BACK TO LOCAL CALLER
FF40:
FF40:      ;-----
FF40:      ;

```

```

FF40:          ; FOLDS INTERP INTO DXXX
FF40:          ; THEN RETURNS TO PASCAL VIA
FF40:          ; RETURN ADDRESS SAVE IN RETL, RETH
FF40:          ;
FF40:          ;-----
FF40:          ;
FF40: 85 0C          GOBACK      STA TT1      ;SAVE A REG
FF42: AD 18BF        LDA RETH
FF45: 48             PHA
FF46: AD 1ABF        LDA RETL
FF49: 48             PHA          ;PUT PASCAL RETURN ADDRESS ON STACK
FF4A: AD 8BC0        LDA 0C08B    ;FOLD INTERP INTO DXXX
FF4B: A5 0C          SKIPIT     LDA TT1    ;RESTORE A REG
FF4F: 60             RTS          ;AND BACK TO PASCAL
FF50:
FF50:          ;-----
FF50:          ;
FF50:          ; PRESERVE OLD IORTS LOCATIONS
FF50:          ;
FF50:          ;-----
FF50:
FF50: 00 00 00 00 00 00 00 00 .ORG 0FF5B
FF5B:
FF5B: 60             IORTS      RTS          ;FIXED RTS OP CODE
FF5F:
FF5F:          ;-----
FF5F:          ;
FF5F:          ; THIS CONCK VECTOR CALLED FROM DOS
FF5F:          ; IF MOVED, CHANGE EQUATI IN DOS
FF5F:          ;
FF5F:          ;-----
FF5F:
FF5F: 4C 8106          JMP CONCK
FF5C:
FF5C:          ;-----
FF5C:          ;
FF5C:          ; THIS CONCK VECTOR USED BY KEYPRESS FUNCTION
FF5C:          ; IF MOVED, CHANGE KEYPRESS FUNCTION
FF5C:          ;
FF5C:          ;-----
FF5C:
FF5C:
FF5C: 20 24FF          JSR SAVERET
FF5F: 20 8106          JSR CONCK
FF62: 4C 40FF          JMP GOBACK
FF65:
FF65:          ;-----
FF65:          ;
FF65:          ; EACH BIOS ROUTINE UNFOLDS DXXX RAM
FF65:          ; THEN DOES ITS BUSINESS, THEN
FF65:          ; FOLDS RAM AGAIN
FF65:          ;
FF65:          ;-----
FF65:
FF65:

```

FF65: AD 83C0	JSTART	LDA 0C083	!UPROT, UNFOLD
FF66: 4C DDD5		JMP START	
FF6B: AD 83C0	JRESET	LDA 0C083	
FF6E: 4C DDD5		JMP RESET	
FF71: AD 88C0	JPASCAL	LDA 0C08B	!FOLD IN INTERP
FF74: 4C 52D1		JMP GOFORIT	
FF77:			
FF018 77FF			
FF77: 20 24FF	QDREAD	JSR SAVERET	
FF78: 20 62B7		JSR CREAD	
FF7B: 4C 40FF		JMP GOBACK	
FF80:			
FF048 80FF			
FF80: 20 24FF	QWRITE	JSR SAVERET	
FF83: 20 D0D7		JSR CURITE	
FF86: 4C 40FF		JMP GOBACK	
FF89:			
FF078 89FF			
FF89: 20 24FF	QINIT	JSR SAVERET	
FF8C: 20 34B7		JSR CINIT	
FF8F: 4C 40FF		JMP GOBACK	
FF92:			
FF0A8 92FF			
FF92: 20 24FF	QWRITE	JSR SAVERET	
FF95: 20 30D8		JSR PURITE	
FF98: 4C 40FF		JMP GOBACK	
FF9B:			
FF0B8 9BFF			
FF9B: 20 24FF	QINIT	JSR SAVERET	
FF9E: 20 83D7		JSR PINIT	
FFA1: 4C 40FF		JMP GOBACK	
FFA4:			
FFA6:			
FF108 A6FF			
FFA6: 20 24FF	QWRITE	JSR SAVERET	
FFA7: 20 3CDD		JSR DWRITE	
FFAA: 4C 40FF		JMP GOBACK	
FFAD:			
FF138 ADFF			
FFAD: 20 24FF	QDREAD	JSR SAVERET	
FFB0: 20 40DD		JSR DREAD	
FFB3: 4C 40FF		JMP GOBACK	
FFB6:			
FF168 B6FF			
FFB6: 20 24FF	QDINIT	JSR SAVERET	
FFB9: 20 04DD		JSR DINIT	
FFBC: 4C 40FF		JMP GOBACK	
FFBF:			
FF198 BFFF			
FFBF: 20 24FF	QDREAD	JSR SAVERET	
FFC2: 20 4ED8		JSR DREAD	
FFC5: 4C 40FF		JMP GOBACK	
FFC8:			
FF1C8 C8FF			
FFC8: 20 24FF	QWRITE	JSR SAVERET	

```

FFC0: 20 09D8      JSR RWRITE
FFCE: 4C 40FF      JMP GOBACK
FFD1:
FF1F: D1FF
FFD1: 20 24FF      ORINIT      JSR SAVERET
FFD4: 20 9CD7      JSR RINIT
FFD7: 4C 40FF      JMP GOBACK
FFDA:
FF22: DAFF
FFDA: 20 24FF      BSWRITE     JSR SAVERET
FFDB: 20 5BFF      JSR IORTS    ;DO NOTHING FOR NOW
FFE0: 4C 40FF      JMP GOBACK
FFE3:
FFE3: ;-----
FFE3: ;
FFE3: ; INIT AND RESET VECTORS
FFE3: ; FOR NOW, NO INTERRUPTS.   MAYBE SOON
FFE3: ;
FFE3: ;-----
FFE3:
FFE3:
FFE3: 00 00 00 00 00 00 00      .ORG OFF6
FFF6:
FFF6:
FFF6: 0100      .WORD 1        ;RELEASE VERSION 1
FFF8: 65FF      .WORD JSTART    ;START POINT FROM BOOT
FFFA: 6BFF      .WORD JRESET    ;INIT VECTOR
FFFC: 6BFF      .WORD JRESET    ;RESET VECTOR
FFFE: 6BFF      .WORD JRESET    ;IRQ VECTOR
0000:
0000:
0000: ;-----
0000: ;
0000: ; THAT'S ALL FOLKS....
0000: ;
0000: ;
0000: ;-----
0000:
0000:
0000:
0000:
0000:
0000:
0000:
0000:
0000:
0000:
0000:
0000:

```

AB - Absolute LB - Label UD - Undefined MC - Macro
 RF - Ref DF - Def PR - Proc FC - Func
 PB - Public PV - Private CS - Consta

ADJLFT	LB DAB8:	ADJOK	LB DAA1:	ADJRT	LB DAAC:	ADJRTS	LB DAAB:	ADJUST	LB DA92:	ADVANCE	LB D9B6:	ADVRTS	LB D9BF:
BASIH	AB 00F1:	BAS1L	AB 00F0:	BAS2H	AB 00F3:	BAS2L	AB 00F2:	BASCAL	LB D9F1:	BELL	LB D957:	BELL2	LB D95E:
BIOS	PR ---:	BREAK	AB BF16:	BSCL2	LB DA02:	BSRET	LB D986:	BUFFER	AB 0200:	BUFOK	LB D717:	BUMP	LB D72C:
BXS1H	AB 0005:	BXS1L	AB 0004:	BXS2H	AB 0007:	BXS2L	AB 0006:	CBUFLEN	AB 004E:	CENIT	LB D791:	CENR	LB D853:
CENW	LB D83F:	CENW2	LB D843:	CH	AB 00F4:	CHECKH	AB 000B:	CHECKL	AB 000A:	CINTT	LB D734:	CINTT2	LB D75F:
CKCTRL	LB D8F3:	CKEXT	LB D725:	CKHX	LB D66B:	CKPAGE	LB D667:	CKPTRH	AB 0009:	CKPTRL	AB 000R:	CLEOL	LB D9A7:
CLEOS	LB D989:	CLNXL	LB D98F:	CLOOP	LB D9A9:	CLRT0	LB D7F4:	CLRT01	LB D7AD:	CLRT02	LB D81C:	CLRT03	LB D7B6:
CNO58YTS	LB D60D:	CNO7BYTS	LB D611:	CNTRL	LB D8F1:	CONCK	LB D6CA:	CONBUF	AB 03B1:	CONCK	LB D681:	CONFLGS	AB BF15:
COPY1	LB DA14:	COPY1	LB D932:	COPY2	LB D93D:	CR	LB D918:	CREAD	LB D762:	CREAD2	LB D771:	CTRLCH	LB D87B:
CTRLRHP	LB D887:	CV	AB 00F5:	CWRTTE	LB D7D0:	DINIT	AB D004:	DMPRTS	LB D733:	DOCTRL	LB D8FC:	DOIT	LB DA8B:
DONECK	LB D71D:	DOPSX	LB D8B5:	DREAD	AB D040:	DRESET	AB D017:	DWRTTE	AB D03C:	ESCAPE	LB D90C:	ESCNT	AB BF12:
FARRT	LB DAB8:	FIRST	AB 00F0:	FORM	LB D97A:	GOBACK	LB FF40:	GOFORIT	AB D152:	GOTOXY	LB D912:	HTDNL	LB DA5E:
HOME	LB D980:	HSHFT	LB DA23:	HTAB	LB DA7E:	ICOM	LB D7A3:	INVERT	LB D9EC:	TORTS	LB FF5B:	ISER	LB D7B0:
JDMCK	LB D6E3:	JTNV	LB D979:	JPASCAL	LB FF71:	JRESET	LB FF6B:	JRET	LB D969:	JSTART	LB FF65:	JUMP1	AB 0002:
JUMP2	AB 0003:	LF	LB D91D:	LF2	LB D923:	LFFLAG	AB BF0F:	LFPASS	LB D33A:	LOP1	LB DA3B:	LOP2	LB DA30:
NCTRLS	AB 000C:	NDBS	LB D973:	NLEFT	AB BF11:	NOCRY	LB D671:	NOPROM	LB D634:	NOTBRK	LB D6F6:	NOTFLUS	LB D706:
NOTFOL	LB D6D5:	NOTK	LB D6AA:	NOTSEB	LB D8DC:	NOTSTOP	LB D6E6:	NOWRP1	LB DA54:	NOWRP2	LB DA63:	NTTAB	LB D4BC:
HXTCRD	LB D5F3:	HXTYP	LB D617:	OFFLFT	LB D9D8:	OFFLINE	LB D84B:	OFFRT	LB D9DE:	OKDEL	LB DA33:	OPPSY	LB D8C7:
PINIT	LB D78B:	PWRTTE	LB D830:	QCINIT	LB FF89:	QCREAD	LB FF77:	QCWRTTE	LB FF80:	QDINIT	LB FF86:	QDREAD	LB FFAD:
QDWRTTE	LB FFA4:	QCWRTTE	LB FFDA:	QPINIT	LB FF9B:	QPWRTTE	LB FF92:	QRTINIT	LB FF01:	QRREAD	LB FFRF:	QRWRITE	LB FFC8:
RANBH	AB BF14:	RANDL	AB BF13:	RCOM	LB D85D:	RESET	LB D5B0:	RETH	AB BF1B:	REN	AB BF1A:	REVL	LB D96A:
RINIT	LB D79C:	RNDINC	LB D687:	RNDOK	LB D68F:	RPTR	AB BF18:	RREAD	LB D84E:	RSER	LB D86C:	RTS7	LB DA22:
RWRTTE	LB D809:	SAVERET	LB FF24:	SCRNODE	AB BF0E:	SCROLL	LB D928:	SECRET	LB D8DB:	SER1	LB D7B9:	SETY	LB D8BC:
SKIPIORT	LB D615:	SKIPIT	LB FF4D:	SKIPSAV	LB FF3D:	SKPCOUT	LB D94B:	SLTTYP	AB BFFB:	START	LB D50D:	STARTUP	LB D664:
STOABV	LB D9B3:	STOR	LB D62E:	STOR2	LB D9F1:	STORE	LB D9C0:	STORE1	LB D9D5:	STORE2	LB D9E9:	SYSCOM	AB 00F8:
TAB2	LB DA85:	TAR3	LB DA89:	TEMP1	AB 00F6:	TEMP2	AB 00F7:	TRYHXT	LB D629:	TSTK80	LB D69A:	TT1	AB 000C:
TT2	AB 000D:	TT3	AB 000E:	UPPER	LB D3EA:	VIDOUT	LB D7E7:	VISLP	LB DA4F:	VOUT2	LB D89F:	WAIT	LB D675:
WAIT2	LB D676:	WAIT3	LB D677:	WCOM	LB D81F:	WPRN	LB D811:	UPTR	AB BF19:	USER	LB D7F7:	XOA	LB D3B7:
YOK	LB D8C9:	ZERLP	LB D5E5:	ZERON	AB 0001:	ZEROL	AB 0000:						

Current minimum space is 8174 words

D64E: ECD9

D6AF: 7EDA

D75D: 89DA

D763: 92DA

D665: 71FF

Assembly complete! 1250 lines

0 Errors flagged on this Assembly



PASCAL - PEEKS & POKES BY DAN SOKAL

This program has been designed to be added to the Pascal SYSTEM.LIBRARY. See section 4.2 in the reference manual for info on the Librarian.

(*SS+,LPRINTER:*)

```
(*****
*                                     *
*          PEEK and POKE           *
*                                     *
*          Dan Sokol    3 Dec 79    *
*                                     *
*                                     *
*****)
```

unit PEEKPOKE; intrinsic code 26;

interface

```
procedure POKE (var ADDR,DATA:integer);
function PEEK (var ADDR: integer): integer;
```

implementation

```
type PA=packed array [0..1] of 0..255;
    MAGIC=record case boolean of
        true : (INT: integer);
        false : (PTR:^PA);
    end;
```

var CHEAT:MAGIC;

```
procedure TEST (var DATA: integer); forward;
```

```
procedure POKE;
begin
    TEST(DATA);
    CHEAT.INT:=ADDR;
    CHEAT.PTR^[0]:=DATA;
end;
```

```

function PEEK;
begin
  CHEAT.INT:=ADDR;
  PEEK:=CHEAT.PTR^[0 ];
end;

procedure TEST;
begin
  DATA:=abs(DATA mod 255);
end;

(*MAIN PROGRAM*)

begin
  (*DUMMY PROGRAM*)
end.

```

SAMPLE SUBROUTINE FOR KEYPRESS

When you use BINDER.CODE to setup your system to an external terminal, or to an 80 column board (like the M & R Sup.R.Terminal), the Apple will not recognize a KEYPRESS from that terminal. This subroutine will correct that - it uses the PEEK & POKE above.

```

(*SI**)
function KEY: boolean;
var keyboard, TEMP:integer
begin
  KEYBOARD:=-16384; TEMP:=PEEK (KEYBOARD);
  if TEMP> 128 then KEY: true else KEY:=false;
end

```


(*I used segment 26*)

(*Format is : *)
(*POKE (addr,data); *)
(*data:=PEEK(addr);

Both addr and data
must be INTEGER variables
(not constants)

To use in a program you
must follow the program
name with :
USES PEEKPOKE;
*)

(* this defines a variant *)
(* record which will map *)
(* to an absolute hardware *)
(* adress in the Apple. *)

(* This procdure assures *)
(* only vaild data will *)
(* get poked :)

TEXT SCREEN MAPPING AND USE

Text Pages

The Apple II has two "pages" of text that it can display. Text page one resides in memory from decimal 1024 to 2047 (\$400-\$7FF). Page two resides from 2048 to 3072 (\$800 to \$C00).

PAGE SWITCHING

It is possible to switch between pages by "POKING" from BASIC. The default setting is page one. If you wish to display page two, the statement "POKE—16299, 0" will display it. "POKE—16300, 0" will return the display to page one. (You must set LOMEM:3072 in order to protect page two from being clobbered.)

"POKING" TEXT

It is possible to place characters on either screen by "POKING" them directly. In order to do this you need to know the memory address of the spot into which you wish to poke the character and the decimal value of the character itself. The following two tables will give you this information, but first a demonstration.

DEMONSTRATION: NO "PRINT" STATEMENTS

```

10 CALL -936
100 POKE 1461, 129: REM "A"
110 POKE 1463, 144: REM "P"
120 POKE 1465, 144: REM "P"
130 POKE 1467, 140: REM "L"
140 POKE 1469, 133: REM "E"
150 POKE 1472, 157: REM "J"
160 POKE 1473, 155: REM THE OTHER J
999 END

```

Text Screen Maps

PAGE ONE	
LINE #	POKE ADDRESS
00	1024
01	1152
02	1280
03	1408
04	1536
05	1664
06	1792
07	1920
08	1064
09	1192
10	1320
11	1448
12	1576
13	1704
14	1832
15	1960
16	1104
17	1232
18	1360
19	1488
20	1616
21	1744
22	1872
23	2000

PAGE TWO	
LINE #	POKE ADDRESS
00	2048
01	2176
02	2304
03	2432
04	2560
05	2688
06	2816
07	2944
08	2088
09	2216
10	2344
11	2472
12	2600
13	2728
14	2856
15	2984
16	2128
17	2256
18	2384
19	2512
20	2640
21	2768
22	2896
23	3024

ADDRESSES SHOWN ARE FOR THE FIRST CHARACTER IN EACH LINE
—LINES ARE 40 CHARACTERS LONG

Character Display Values

CHAR	NORMAL	INVERSE	FLASH	CHAR	NORMAL	INVERSE	FLASH
@	128	0	64	!	161	33	97
A	129	1	65	"	162	34	98
B	130	2	66	#	163	35	99
C	131	3	67	\$	164	36	100
D	132	4	68	%	165	37	101
E	133	5	69	&	166	38	102
F	134	6	70	'	167	39	103
G	135	7	71	(168	40	104
H	136	8	72)	169	41	105
I	137	9	73	*	170	42	106
J	138	10	74	+	171	43	107
K	139	11	75	,	172	44	108
L	140	12	76	-	173	45	109
M	141	13	77	.	174	46	110
N	142	14	78	/	175	47	111
O	143	15	79	0	176	48	112
P	144	16	80	1	177	49	113
Q	145	17	81	2	178	50	114
R	146	18	82	3	179	51	115
S	147	19	83	4	180	52	116
T	148	20	84	5	181	53	117
U	149	21	85	6	182	54	118
V	150	22	86	7	183	55	119
W	151	23	87	8	184	56	120
X	152	24	88	9	185	57	121
Y	153	25	89	:	186	58	122
Z	154	26	90	;	187	59	123
[155	27	91	<	188	60	124
/	156	28	92	.	189	61	125
]	157	29	93	>	190	62	126
<	158	30	94	?	191	63	127
-	159	31	95				
SPACE	160	32	96				

How To Use It

LOOK!
NO "PRINT" STATEMENTS!

```

100 CALL -936: POKE 74,0: POKE 75,12: POKE 204,0: POKE 205
    ,12: REM SET LOMEM & VAR PTR
110 POKE 1024,144: POKE 1025,129
    : POKE 1026,135: POKE 1027,
    133: REM POKE "PAGE" IN10 PG 1
120 POKE 34,2: VTAB 3: LIST
130 TRACE: DIM C$(30)
140 C$="800<400.7FFM E8BAG"
150 FOR CHR=1 TO LEN(C$)
160 POKE 511+CHR, ASC(C$(CHR))
170 NEXT CHR: CALL -144
180 POKE 2053,50: POKE 1029,49
190 FOR T=1 TO 5
200 POKE -16299,0: REM PAGE 2
210 FOR D=1 TO 100: NEXT D
220 POKE -16300,0: REM PAGE 1
230 FOR D=1 TO 100: NEXT D
240 NEXT T: NOTRACE: CALL -936
    LIST: POKE 34,0: END

```

Line 100

—Clears the screen and sets LOMEM to 3072.

Line 110

—Pokes the word "page" into the upper left corner of page one (character by character).

Line 120

—Sets scrolling window and lists the program.

Line 130

—Sets trace mode and dimensions C\$.

Line 140

—Sets C\$ equal to a couple of monitor commands. "800<400.7FFM" is a command which moves the contents of page one to page two display memory. "E8BAG" tells integer BASIC to continue where it left off.

Lines 150-170

—Poke the commands (C\$) into the input buffer, character at a time, then call a monitor routine which executes the contents of the buffer (The input buffer is from 512 to 767 decimal or \$200 to \$2FF Hex).

Line 180

—Pokes an inverse "2" into page two and an inverse "1" into page one.

Lines 190-240

—Switch from page one to page two a few times, then exit from the program with page one set and the program listed.



APPLICATIONS NOTE FOR PASCAL

PASCAL LONG INTEGER FIX

An error in the implementation of long integers in the Apple PASCAL language system results in a stack crash during a compare operation.

This program is designed to repair the library module LONGINTEGER. To use, type the program in, then compile and execute. All libraries should be updated with this program. Save the text for possible future use.

```

{$I-}
PROGRAM FIXCOMPARE;

TYPE DISKINFO = RECORD
    DADDR:INTEGER;
    LENG: INTEGER
    END;

NAME = PACKED ARRAY[1..8] OF CHAR;

SEGDIC = RECORD
    DINFO: ARRAY[0..15] OF DISKINFO;
    SEGNAME:ARRAY[0..15] OF NAME;
    FILLER: ARRAY[1,,416] OF INTEGER
    END;

BLOCK = PACKED ARRAY[0..511] OF 0..255;

VAR I,J:INTEGER;
    NOTFOUND:BOOLEAN;
    F:FILE;
    S:STRING;
    BLOCKZERO:SEGDIC;
    DATA:BLOCK;

PROCEDURE READERROR;
BEGIN
    WRITE('BAD BLOCK IN LIBRARY'); EXIT(PROGRAM)
END;

BEGIN
    NOTFOUND:=TRUE;
    REPEAT
        WRITE('NAME OF LIBRARY FILE:'); READLN(S);
        RESET(F,S);
    UNTIL EOF OR (IORESULT=0);
    IF BLOCKREAD(F,BLOCKZERO,1,0) <> 1 THEN READERROR;
    FOR I := 0 TO 15 DO
        BEGIN
            IF BLOCKZERO.SEGNAME[I] = 'LONGINTI' THEN
                BEGIN
                    NOTFOUND := FALSE;
                    J := BLOCKZERO.DINFO[I].DADDR + 1;
                    IF BLOCKREAD(F,DATA,1,J)<>1 THEN READERROR;
                    IF DATA[495]=244 THEN
                        IF DATA[494]=208 THEN
                            BEGIN
                                WRITELN('LONG INTEGER PATCH BEING MADE');
                                DATA[494]:=240;
                                IF BLOCKWRITE(F,DATA,1,J) = 1 THEN
                                    WRITELN('PATCH COMPLETE')
                                ELSE
                                    WRITELN('ERROR WHILE WRITING PATCH, SEGMENT ',I);
                            END
                        END
                    END
                END
            END
        END
    END

```

```

ELSE
  IF DATA[494]=240 THEN
    WRITELN('SEGMENT ',I,
            ' - LONG INTEGER UNIT HAS ALREADY BEEN FIXED')
  ELSE
    WRITELN('CAN''T RECOGNIZE LONG INTEGER UNIT IN SEGMENT ',I)
  ELSE
    WRITELN('CAN''T RECOGNIZE LONG INTEGER UNIT IN SEGMENT ',I)
  END
END;
IF NOTFOUND THEN WRITELN('NO LONG INTEGER UNIT FOUND')
ELSE WRITELN('PROGRAM COMPLETE');
END.

```



PASCAL HI-RES LOAD/SAVE TO DISK

This demo program creates a hi-res picture in PASCAL, then saves it to disk. It is then reloaded and displayed.

The "Uses Turtlegraphics" statement allocates space for the hi-res screen, and should be referenced even if Turtlegraphics are not actually used. Note that the "Close (F,Lock)" closes the file and places it permanently into the volume directory.



PASCAL UNITS

Modular programming means the separation of procedures and functions, or groups of them, from the main program. Source language modules are called **UNITs**, and are incorporated in libraries for use with Pascal programs. Units may consist of procedures, functions, or a combination of these, in Pascal and in assembly language.

Separate compilation has several advantages in the development of any program, because it allows you to approach the task as a group of smaller tasks which are linked together in a logical manner. The host program must contain a **USES** statement to utilize routines from the **UNIT**.

There are two principal kinds of **UNITs**: Regular **UNITs**, and Intrinsic **UNITs**. When a host program **USES** a Regular **UNIT**, the **UNIT's** code is inserted into the host program's codefile by the Linker. This needs to be done only once unless the **UNIT** is modified and recompiled; then it must be relinked into the host program.

When a host program **USES** an Intrinsic **UNIT**, the **UNIT's** code remains in the library file and is automatically loaded into memory when the host program is executed. This keeps the size of the host program's codefile down; it also allows the **UNIT** to be modified and recompiled without the need to relink. If the **UNIT** resides in the **SYSTEM.LIBRARY**, the Linker will be called automatically. Otherwise, you must explicitly invoke the Linker.

Pages 187-195 in the Pascal Reference Manual explain the syntax and structure of **UNITs**.

Separate **UNITs** do not work in the current Pascal implementation.



**INTERNATIONAL
APPLE CORE**

TM

P. O. BOX 976, DALY CITY, CALIFORNIA 94017 USA

APNOTE

DOS 3.2 DEMONSTRATION PROGRAMS

The following programs demonstrate the various features of DOS 3.2. They are written for Applesoft and some of the programs will be difficult to convert to Integer Basic.

]LIST

```
100 REM TEST MON
110 LET D$ = CHR$(4)
120 LET S$(1) = "NO MONITOR"
130 LET S$(2) = "MON C"
140 LET S$(3) = "MON I"
150 LET S$(4) = "MON O"
160 LET S$(5) = "MON C,I"
170 LET S$(6) = "MON C,O"
180 LET S$(7) = "MON I,O"
190 LET S$(8) = "MON C,I,O"
200 LET S$(8) = "MON C,I,O"
210 HOME
220 VTAB 1: HTAB 16: PRINT "TEST MON"
230 FOR L = 1 TO 8
240 VTAB L + 4: HTAB 5: PRINT "<"L"> S$(L)
250 NEXT
260 VTAB 15: PRINT "ENTER A NUMBER (1-7): "
270 PRINT : PRINT "PRESS 'ESC' TO LEAVE TEST MON"
280 VTAB 15: HTAB 23: GET G$
290 IF ASC (G$) = 27 THEN HOME : END
300 LET G = VAL (G$)
310 IF G < 1 OR G > 8 THEN 280
320 PRINT
330 PRINT D$"NOMON I,O,C"
340 HOME
350 PRINT TAB( 20 - LEN (S$(G)) / 2);S$(G)
360 PRINT : PRINT
370 IF G = 1 THEN 390
380 PRINT D$;S$(G)
390 PRINT TAB( 11);"WRITING TO THE DISK": PRINT
400 PRINT D$"OPEN TESTER"
410 PRINT D$"WRITE TESTER"
420 PRINT "THIS IS THE DATA TO THE DISK"
430 PRINT D$"CLOSE TESTER"
440 VTAB 12: PRINT TAB( 10);"READING FROM THE DISK": PRINT
450 PRINT D$"OPEN TESTER"
460 PRINT D$"READ TESTER"
470 INPUT A$
480 PRINT D$"CLOSE TESTER"
490 PRINT D$"DELETE TESTER"
500 VTAB 20: PRINT "PRESS ANY KEY FOR THE MENU ";
510 POKE - 16368,0
520 GET A$
530 GOTO 110
```

]

```

0  REM MAKE TEXT
10  DIM A$(100):I = 0
20  D$ = CHR$(4): REM CTRL-D
25  TEXT : HOME
30  PRINT "THIS PROGRAM LETS YOU WRITE TEXT FILES."
34  PRINT "YOU MAY TYPE ONE STRING AT A TIME."
38  PRINT "A STRING MAY HAVE UP TO 239 CHARACTERS."
40  PRINT : PRINT "(PRESS THE RETURN KEY TO QUIT)"
50  POKE 34,7: VTAB 7
52  I = I + 1
54  PRINT "TYPE STRING #";I;": ";
56  INPUT A$(I)
58  IF A$(I) < > "" GOTO 52
60  PRINT
62  INPUT "FILE NAME TO STORE ? ";N$
65  IF N$ < > "" GOTO 70
67  PRINT : PRINT "FILE NOT SAVED": GOTO 140
70  PRINT D$;"OPEN ";N$
80  PRINT D$;"WRITE ";N$
90  PRINT I - 1
100 FOR J = 1 TO I - 1
110 PRINT A$(J)
120 NEXT J
130 PRINT D$;"CLOSE ";N$
135 PRINT : PRINT "TEXT FILE SAVED"
140 POKE 34,0: END

```

```
0  REM GET TEXT
5  TEXT : HOME
10 D$ = CHR$ (4): REM  CTRL D
12  PRINT "THIS PROGRAM RETRIEVES TEXT FILES"
14  PRINT "CREATED BY THE 'MAKE TEXT' PROGRAM."
16  PRINT : PRINT "MON C,I,O IS IN EFFECT."
18  PRINT
20  INPUT "NAME OF TEXT FILE? ";Z$
22  PRINT D$;"MON C,I,O"
24  PRINT
30  PRINT D$;"OPEN ";Z$
40  PRINT D$;"READ ";Z$
50  INPUT I
55  DIM A$(I)
60  FOR J = 1 TO I
70 : INPUT A$(J)
80  NEXT J
90  PRINT D$;"CLOSE ";Z$
100 PRINT D$;"NOMON C,I,O"
```

```

0  REM MAKE RANDOM
1  REM FOR USE WITH 'RANDOM'
10 D$ = CHR$(4)
20 DIM N$(9),BL(9),BW(9),ST(9)
30 FOR I = 1 TO 9
40 READ N$(I),BL(I),BW(I),ST(I)
45 NEXT I
50 PRINT D$;"OPEN APPLE PROMS,L40"
55 PRINT D$;"DELETE APPLE PROMS"
65 PRINT D$;"OPEN APPLE PROMS,L40"
70 FOR I = 1 TO 9: PRINT D$;"WRITE APPLE PROMS,R";I
80 PRINT N$(I): PRINT BL(I): PRINT BW(I): PRINT ST(I)
90 NEXT I
100 PRINT D$;"CLOSE APPLE PROMS"
110 PRINT : PRINT "FILE 'APPLE PROMS' MADE": END
1000 DATA PARALLEL PRINT,256,8,500,COMMUNICATIONS,256,8,1250,(NOT AVAILA
    BLE),256,8,0,(NOT AVAILABLE),256,8,0,DISK BOOT,256,8,432
1010 DATA STATE MACHINE,256,8,460,SERIAL PRINTER1,256,8,878,SERIAL PRINT
    ER2,512,8,741,CENTRONICS,256,8,1290

```

]

```

0 REM RANDOM
5 D$ = CHR$ (4)
10 PRINT D$;"NOMON C,I,O"
15 TEXT : HOME
20 LET OP$ = D$ + "OPEN "
30 LET CL$ = D$ + "CLOSE "
40 LET RD$ = D$ + "READ "
50 LET WR$ = D$ + "WRITE "
60 LET FL$ = "APPLE PROMS"
70 PRINT OP$;FL$;" ,L40"
80 GOSUB 390
90 ON Q GOTO 100,180,480
100 GOSUB 330
110 FOR R = R1 TO R2
120 PRINT RD$;FL$;" ,R";R
130 INPUT N$,BL,BW,ST
140 PRINT " ";R; TAB( 8);N$; TAB( 24);BL; TAB( 32);ST
150 NEXT R
160 PRINT D$
170 GOTO 310
180 GOSUB 330
190 LET T = 7: FOR R = R1 TO R2:T = T + 1
200 PRINT RD$;FL$;" ,R";R: INPUT N$,BL,BW,ST: PRINT D$
210 VTAB (T): PRINT " ";R; TAB( 8);N$;: HTAB (7): INPUT Q$
220 IF LEN (Q$) > 15 THEN 200
230 IF LEN (Q$) < > 0 THEN N$ = Q$
250 VTAB (T): HTAB (24): PRINT BL;: HTAB (23): INPUT Q$: IF LEN (Q$) <
    > 0 THEN BL = VAL (Q$)
270 VTAB (T): HTAB (32): PRINT ST;: HTAB (31): INPUT Q$: IF LEN (Q$) >
    0 THEN ST = VAL (Q$)
280 VTAB (T): PRINT " ";R; TAB( 8);N$; TAB( 24);BL; TAB( 32);ST;" "
290 PRINT WR$;FL$;" ,R";R: PRINT N$;" ,";BL;" ,";BW;" ,";ST
300 PRINT D$: NEXT R
310 VTAB (23): PRINT "PRESS THE RETURN KEY TO CONTINUE.";: GET Q$
320 GOTO 80
330 PRINT : INPUT "PART NUMBER 1-9 (0=ALL) ";Q$
340 PRINT Q$:Q = VAL (Q$): IF (Q < 1 OR Q > 9) AND Q$ < > "0" THEN PRINT
    CHR$ (7);: GOTO 330
350 LET R1 = Q:R2 = Q: IF Q = 0 THEN R1 = 1:R2 = 9
360 HOME : VTAB (5)
370 PRINT "PART# NAME SIZE IN STOCK -----"
    ----- "
380 RETURN
390 HOME : PRINT TAB( 12);FL$: VTAB (10)
400 PRINT "COMMAND","NUMBER"
410 PRINT "-----","-----"
420 PRINT "LIST"," 1"
430 PRINT "CHANGE"," 2"
440 PRINT "EXIT"," 3"
450 PRINT : INPUT "CHOOSE NUMBER (1 - 3) ";Q$:Q = VAL (Q$)
460 IF Q > 0 AND Q < 4 THEN RETURN
470 VTAB (15): PRINT CHR$ (7);: GOTO 450
480 PRINT CL$;FL$
490 HOME : END
500 REM DEMONSTRATION OF RANDOM ACCESS

```

```

0  REM EXEC DEMO
100 Q$ = CHR$ (34): REM 34 IS THE ASCII CODE FOR A QUOTATION MARK
110 TEXT : HOME : VTAB 2: HTAB 12
120 INVERSE : PRINT "<< EXEC DEMO >>": NORMAL : PRINT
130 PRINT "THIS PROGRAM CREATES A SEQUENTIAL TEXT"
140 PRINT "FILE NAMED "Q$DO'ER"Q$" CONTAINING SEVERAL"
150 PRINT "STRINGS, EACH A LEGAL APPLE II COMMAND."
160 PRINT "WHEN YOU TYPE"
170 PRINT : PRINT "EXEC DO'ER"
180 PRINT : PRINT "THEN THE COMMANDS IN FILE DO'ER TAKE"
190 PRINT "CONTROL OF YOUR COMPUTER. EACH COMMAND"
200 PRINT "WILL BE EXECUTED JUST AS IF IT HAD BEEN"
210 PRINT "TYPED AT THE KEYBOARD. THE DOS MANUAL"
220 PRINT "DESCRIBES THE PROGRAM IN MORE DETAIL."
230 PRINT : HTAB 10
240 INVERSE : PRINT "<< HAPPY EXECUTING >>": NORMAL : PRINT
250 PRINT "PRESS THE SPACE BAR TO MAKE THIS"
260 PRINT "PROGRAM CREATE THE FILE DO'ER."
270 PRINT
280 PRINT "IF YOU WISH TO STOP THIS PROGRAM NOW,"
285 PRINT "YOU MAY PRESS THE ESC KEY."
288 REM  END INSTRUCTIONS AND WAIT FOR KEY TO BE PRESSED.

290 GET A$: IF A$ = CHR$ (27) THEN  END : REM  ESC KEY PRESSED
300 IF A$ = CHR$ (32) THEN 320: REM  SPACE BAR PRESSED
310 PRINT CHR$ (7);: GOTO 290: REM  BEEP AND TRY AGAIN

320 HOME : PRINT : REM  PROGRAM STARTS HERE
330 D$ = CHR$ (4): REM CTRL-D
340 PRINT D$"MON C,I,O"
350 PRINT D$"OPEN DO'ER"
360 PRINT D$"WRITE DO'ER"
370 PRINT "FP"
380 PRINT "MON C,I,O"
385 PRINT "REM HERE IS A PROGRAM"
390 PRINT
395 PRINT "100 TEXT:HOME:VTAB 5"
400 PRINT "110 PRINT"Q$"HERE'S A NEW PROGRAM"Q$
410 PRINT "120 END"
415 PRINT
420 PRINT "SAVE NEW PROGRAM!!"
425 PRINT
430 PRINT "LIST : REM  NEW PROGRAM!!"
435 PRINT "REM PAUSE TO LOOK AT LISTING"
440 PRINT "FOR X=1 TO 8000: NEXT X"
450 PRINT "INT"
460 PRINT "MON C,I,O"
470 PRINT "LOAD COLOR DEMO"
480 PRINT "LIST"
490 PRINT "FP"
495 PRINT "PRINT"Q$"PAUSE TO LOOK AT LISTING"Q$
500 PRINT "FOR X=1 TO 4000:NEXT X"
505 PRINT "MON C,I,O"
510 PRINT "CALL -155 : REM  JUMPS TO MONITOR"
520 PRINT "800.820 I 821.840"

```


530 PRINT "FP"
535 PRINT "PRINT"Q\$"PAUSE TO LOOK AT MONITOR LISTING"Q\$
540 PRINT "FOR X=1 TO 4000:NEXT X"
545 PRINT "MON C,I,O"
550 PRINT "CATALOG"
560 PRINT "RUN NEW PROGRAM!!"
565 PRINT
570 PRINT "115 PRINT"Q\$"WE CAN EVEN CHANGE IT"Q\$
575 PRINT
580 PRINT "LIST : REM EVEN MORE RECENT PROGRAM!!"
585 PRINT "REM PAUSE TO LOOK AT NEW LISTING"
590 PRINT "FOR X=1 TO 8000:NEXT X"
600 PRINT "SAVE EVEN MORE RECENT PROGRAM!!"
610 PRINT "DELETE NEW PROGRAM!!"
620 PRINT "CATALOG"
630 PRINT D\$"CLOSE DO'ER"
640 PRINT : PRINT : INVERSE : PRINT "IT'S DONE!!!": NORMAL
650 PRINT
660 PRINT "YOUR APPLE'S READY TO DO'ER IT'S THING!"
670 PRINT "ALL YOU HAVE TO DO IS TYPE"
680 PRINT "EXEC DO'ER"
690 PRINT "PRESS THE RETURN KEY, AND SIT BACK."